Politecnico di Torino

Masters degree in Electronics Engineering

Laboratory :01
Design and implementation of a digital filter (FIR)

Integrated system architecture

Group 29

Hamza Sadiq

November 21, 2021

Abstract

The goal of this laboratory session is to design, implement, and improve using universal techniques, a digital hardware filter.
The filter under analysis is a finite impulse response (FIR) one, low-pass, 14 bit of width, order equal to 10, and cut-off frequency of 2 kHz, developed using the direct form.
To start the design a MATLAB script is written with two purposes: first a txt file is written containing the samples of a input test signal composed by only two harmonics: one lower than the cut-off frequency, and one higher; then the filter is simulated using a built-in MATLAB function in order to get the multiplication coefficients and a file containing the output samples (only the lower harmonic must remain). With these it was possible to implement the filter first in a C program, and once the results fit the MATLAB ones, in a VHDL design. The latter was then improved using 3-level unfolding and pipelining.

As final step, both the basic and the advanced design are synthesized and place-and-routed.

Contents

List of Figures

List of Tables

List of Listings

CHAPTER 1

Reference model development

The first step consists in generating the input samples and to simulate the filter in MATLAB environment.In listing A.1 there is the MATLAB function, which in turn uses the built-in function *fir1(N,f0)*, aimsto get the multiplication coefficients and to plot the filter response that is reported in figure 1.1.

This user-defined function is then used in the script of listing A.2, which has the two following tasks:

- It generates the file of listing A.4. This file contains a sequence of samples corresponding to a signal composed by the mean of two sinewaves of 500 Hz (in pass-band) and 4.5 kHz (out pass-band) and sampled with a frequency of 10 kHz.
- It simulates the filter with the given specification and giving to the input the samples obtained in the previous point, another file is written with the corresponding output samples (listing A.6). In figure 1.2 you can see both the input samples (and also the two single components) and the outputs samples.

These files will be useful to test first the C program implementation and then the VHDL design. In particular the input samples will be the same for both, and the outputs will be compared to the ones obtained with the matlab simulation: if the output samples are equal, the filter work correctly.

The second step is the C program implementation, the source code is listed in listing A.3. The program operation is pretty simple: it receives as parameters two files, the first is where the input samples are found, and the second is where the output samples are written.

As can be seen in the output file (listing A.7) the results are very close to the ideal ones obtained in MATLAB environment, with a small error due to the quantization, so the program model work good.

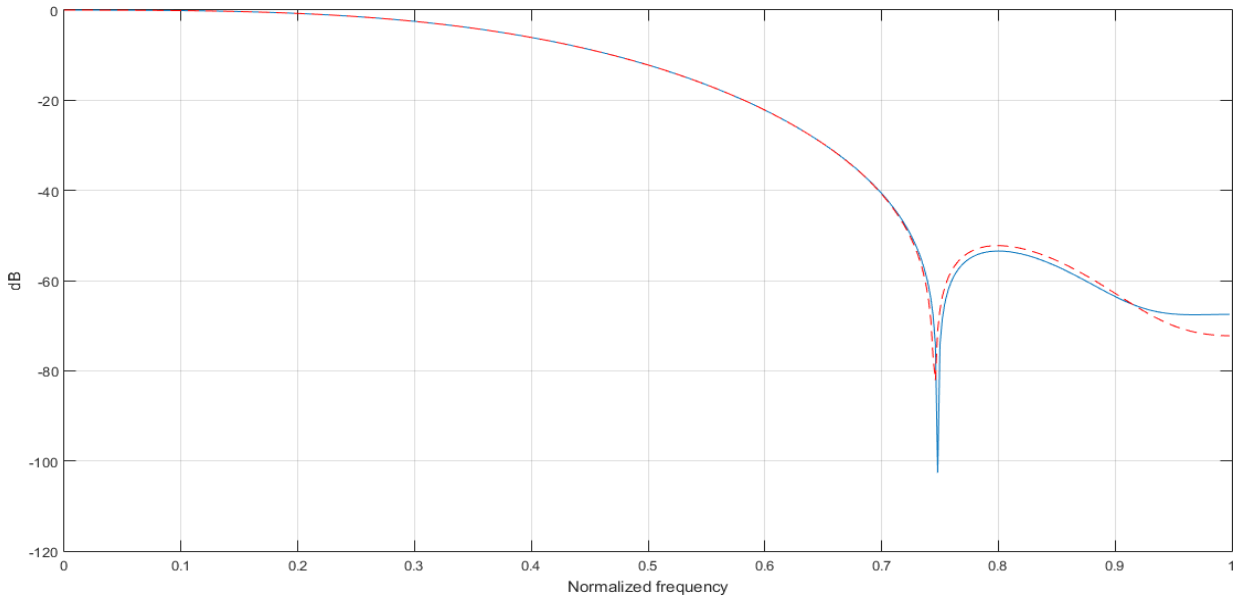


Figure 1.1: Filter response obtained in MATLAB environment. The blue curve is the ideal response, thered curve is the response of the quantized filter.

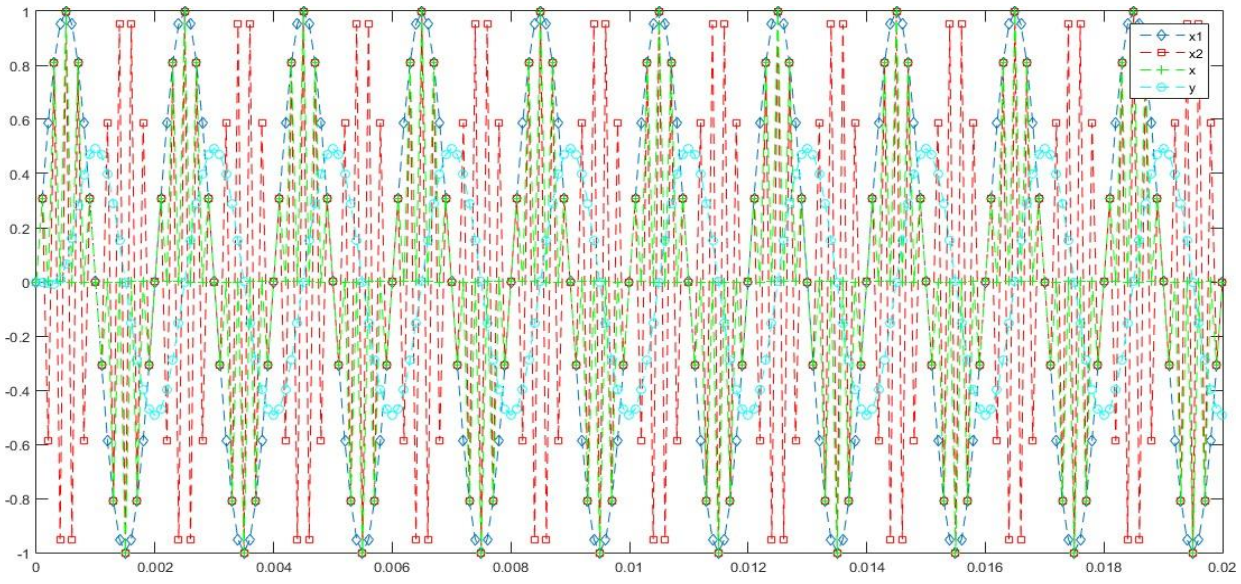


Figure 1.2: Blue: samples of the first input component. Red: samples of the second input component. Green: resulting input samples. Cyano: output samples.

CHAPTER 2

VLSI implementation

The VLSI design can be divided into six steps:

1. HDL description of the filter;
2. Testing;
3. Synthesis;
4. Post-synthesis simulation and Switching-activity estimation;
5. Place-and-route;
6. Post-place-and-route simulation and Switching-activity estimation.

2.1 HDL description

The filter is entirely descripted in VHDL language. The top-level entity is illustrated in figure 2.1,the meaning of each port is the following:

- DIN is the data input.
- DOUT is the data output.
- VIN is high when the input data is valid;
- VOUT is high when the output data is valid;
- $b_0 \ldots b_{n_b}$ are the multiplication coefficients.
- CLK is the positive edge triggered input clock.
- RST n is the active-low synchronous reset.

The internal block diagram is shown in figure 2.2. Inside it there are only four types of sub-block: multiplier, adder, register, and multiplexer. All are described in a behavioural way, the first two using the arithmetic operators in order to let the synthesizer choose the better implementation, the third using a sequential process, and the latter using a select statement. Furthermore, the register reset is synchronous and active-low, while the register enable is active-high.

The multiplexer before the data output has the task to send to the output the maximum (or the minimum) value containable in 14 bits, if the computed sample is too high (or to low) to be represented in such number of bits.

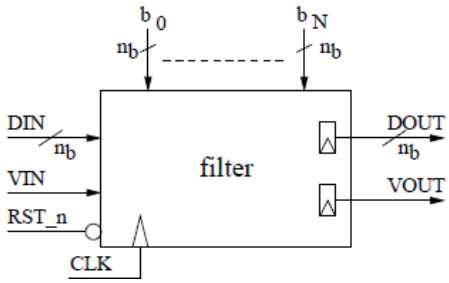


Figure 2.1: Top level entity diagram.

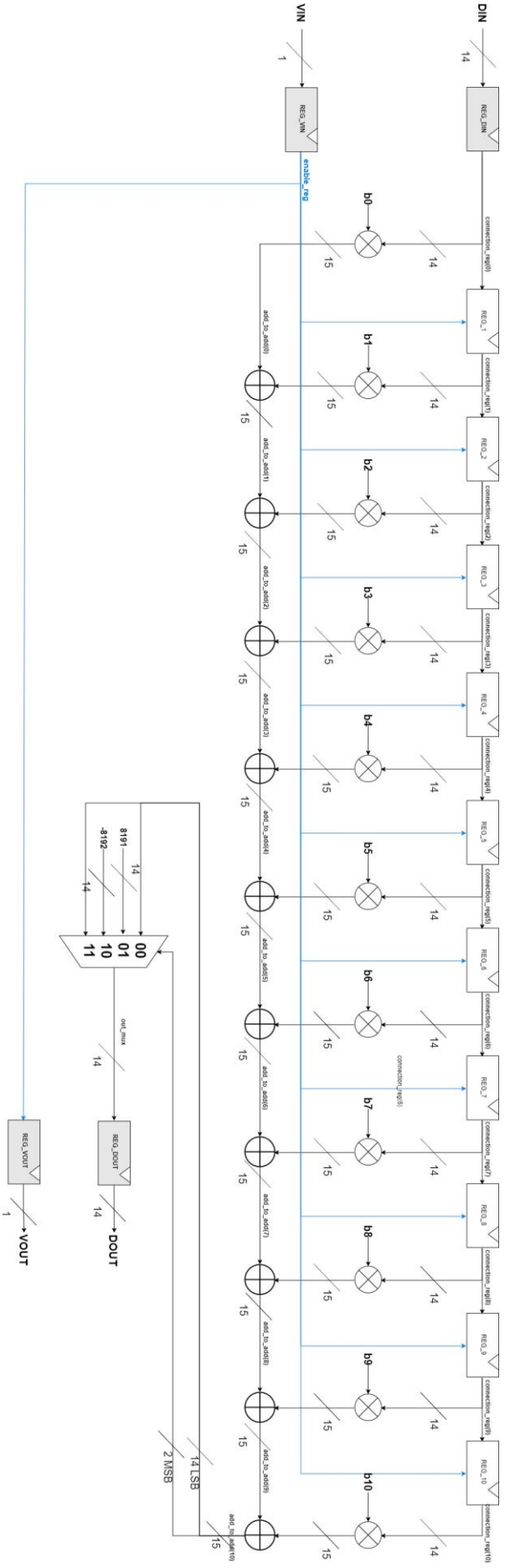


Figure 2.2: Filter internal diagram.

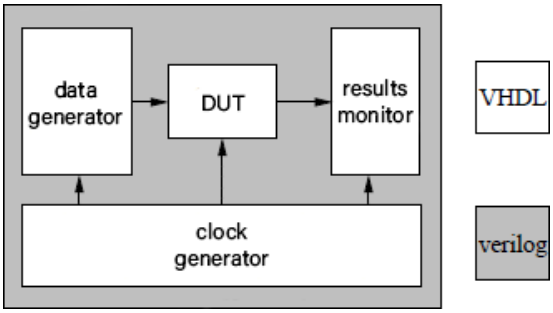


Figure 2.3: Testbench connections diagram.

2.2 Testbench

Also, the testbench is split into more blocks:

1. The clock generator.
2. The data generator, which reads from the samples file.
3. The Results monitor, which writes the outputs in a file.

All these sub-blocks are developed in VHDL, but the wrapper file is a Verilog one, in figure 2.3 you can see the complete block diagram of the testbench, in which the used languages are also underline.

2.2.1 Clock generator

This module, in addition to generate the clock for all the other modules, sends a short negative pulse on the reset, in order to initialize correctly the filter and to make start the data generator.

Furthermore, if it receives from the data generator a signal that says the simulation is over, the clock generation stops. This will be useful when the switching activity has to be evaluated, because in this way all the sequential elements no more switch when there are no more input samples.

2.2.2 Data generator

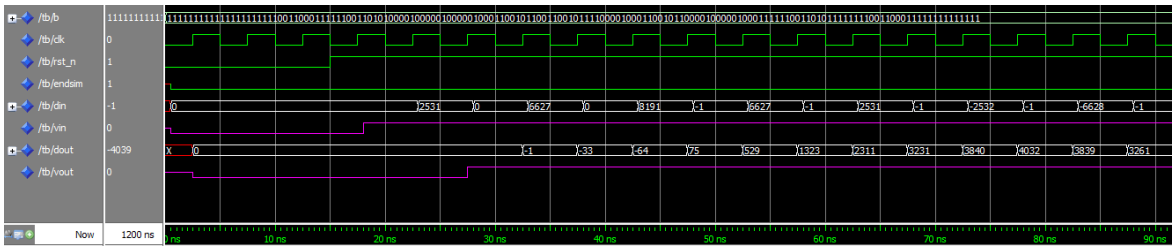
This file has the task to read the samples generated by the MATLAB script and to give them to the filter under test in a sequential order according to the received clock. As just mentioned, a flag read by the clock generator is raised when all the data are given to the filter. In order to test also the validation input and validation output signals, at a certain time the first is lowered and as consequence also the validation output has to become low.

2.2.3 Results monitor

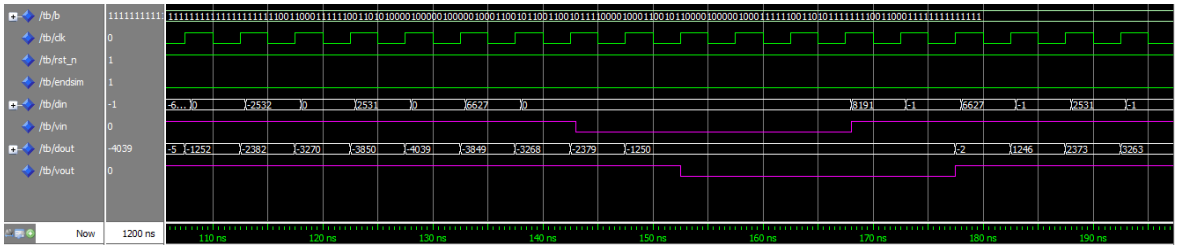
This component is the one that allow to verify the design by the methodology explained in chapter 1, in fact all the samples output by the filter are sequentially filled in a file. Clearly, to correctly test also the *vout*, this happens only when this signal is active.

2.2.4 ModelSim simulation

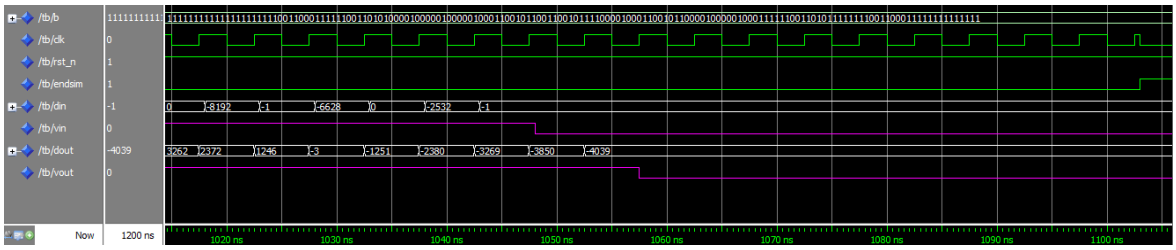
The testbench described above is then runner using Mentor ModelSim.



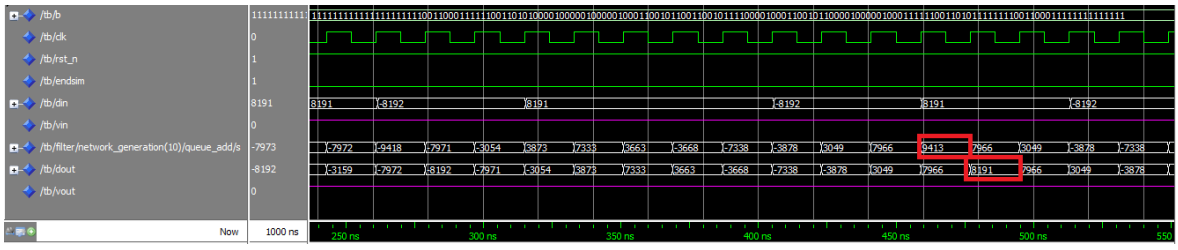
(a) Filter's behaviour at the starting point. Validation signals are in violet to easy distinguish them from the other signals.



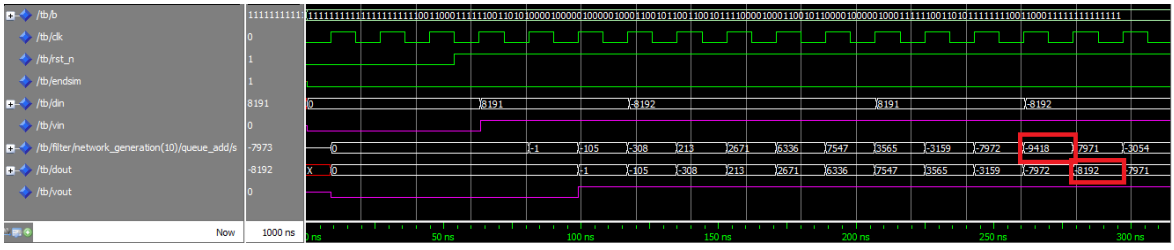
(b) Filter's behaviour when the validation input goes to 0: at around 140 ns *it stay down for five clock cycles, at around* 180 ns *all restart normally.*



(c) Filter's behaviour at the end of the process.



(d) Saturation to the maximum, at around 470 ns.



(e) Saturation to the minimum, at around 270 ns.

Figure 2.4: ModelSim simulations.

In figure 2.4a it is possible to see the reset procedure, which takes three clock cycles in our case, and the starting of the computation of the first samples. The latency between *din* and *dout* is two clock cycles because two registers are introduced between them, the same for *vin* and *vout*.

In figure 2.4b you can observe that when *vin* goes to zero, the filter doesn't allow the entering of new samples until *vin* return to one.

In figure 2.4c there is the computation of the last samples and the *ENDSIM* signal behaviour, already explained above.

In conclusion, in figure 2.4d and in figure 2.4e the saturations to the maximum and to the minimum value are underlined. In order to test properly this saturation process, the samples provided to the filter are changed with the ones in listing A.5 so that the maximum and the minimum achievable numbers inside the filter are obtained.

2.3 Synthesis

The synthesis is performed by Synopsys Design Compiler using the TCL script of listing B.11. Schematically, this script execute the following tasks:

1. Load all the needed files;
2. Perform a formal verification of such files;
3. Synthesise the top level files giving a maximum clock frequency constraint (clock affected by an uncertainty of 0.07 ns to take into account the jitter) and a maximum delay for input and output;
4. Generate SDC, SDF and the synthesised netlist in verilog format;
5. Generate timing and area reports.

First of all the script was runned setting a clock with a period of 0 ns in order to find the maximum frequency. This gave as a result a negative slack of 3.77 ns corresponding to a frequency of 265.25 MHz. As second step, another synthesis was performed using this timing result as constraint. The result of this second synthesis had still a slack violation, so other synthesis was carried out until the maximum frequency without slack violations was found, this turned out to be 224.72 MHz. With this value we finally could make the final synthesis forcing a clock period four times larger than the minimum, so with a frequency of 56.17 MHz. In table 2.1 you can see the area reports with these last two frequencies.

In order to evaluate a more accurate power consumption, Mentor Modelsim and Synposys Design Compiler was jointly used in order to get a switching-activity-based estimation. The result of this operation is reported in table 2.2.

Max freq. [MHz]	**Comb. area** [μm^2]	**Non comb. area** [μm^2]	**Total area** [μm^2]
227.72	9228.604054	777.783973	10006.388027
59.17	9047.990060	768.739972	9816.730033

Table 2.1: Post synthesis area report.

	Internal Power	**Switching Power**	**Leakage Power**	**Total Power**
µW	416.61	374.63	200.78	998.02
%	44	37.76	20.24	100

Table 2.2: Post synthesis switching-activity-based power consumption.

2.4 Place-and-route

Also this phase was done using a script, shown in listing B.15. The sequence of operations is the following:

1. Load the synthesized netlist, the cell library, and the configuration files;
2. Specify the size of the silicon die and its core;
3. Add a power ring for VDD and one for VSS;
4. Prepare the horizontal wires for the cells power supply;
5. Place the cells on the core;
6. Optimise the design.
7. Place filler;
8. Route all the connections.
9. Optimise again;
10. Save the design and the snapshot;
11. Extract the parasitic impedences;
12. Verify the design;
13. Write reports, post-place-and-route netlist, and SDF.

In figure 2.5 the overview of the floorplan is illustrated while the area given by the report is $9820.2\,\mu m^2$.

At this point the obtained netlist was again simulated and the switching-activity evaluated. A summary report of the power consumption obtained is in table 2.3.

From these data we can notice that after the place-and-route procedure the power consumption is grew by about 50%, this may be due both to the insertion of buffers in order to respect the timing constraint, and to the non-idealities of the connections that in the synthesis phase are not considered.

	Internal Power	Switching Power	Leakage Power	Total Power
mW	0.721	0.573	0.202	1.496
%	48.22	38.29	13.49	100

Table 2.3: Post place-and-route switching-activity-based power consumption.

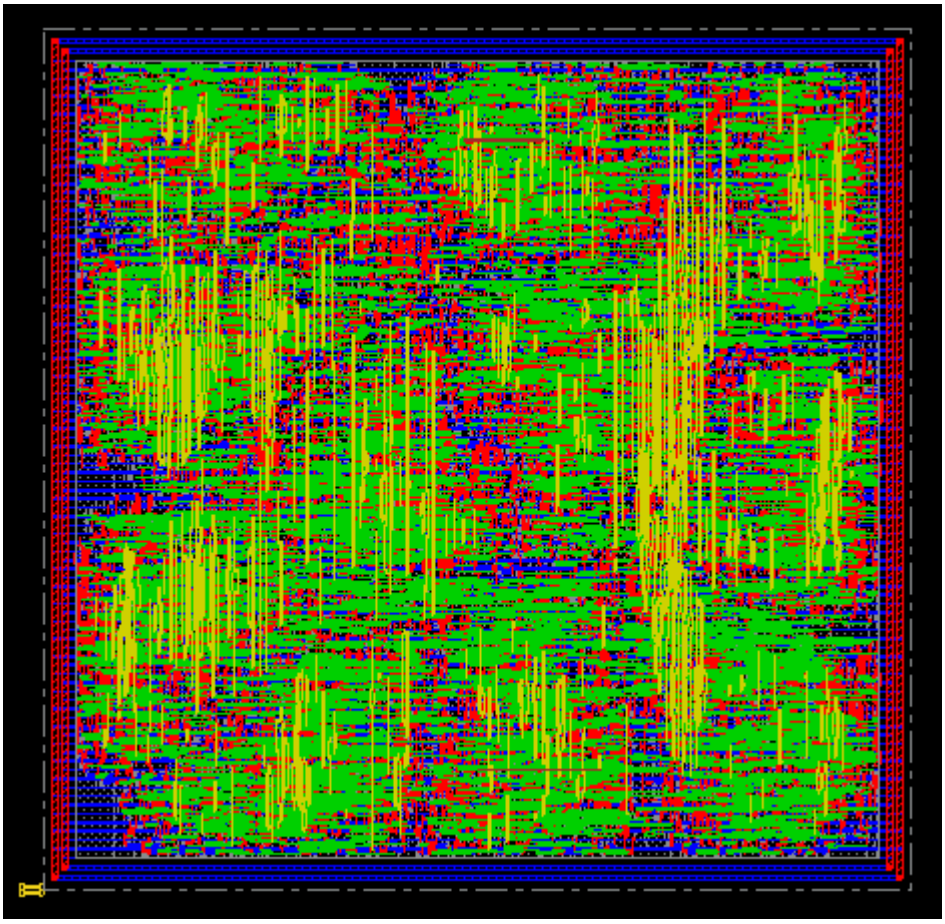


Figure 2.5: Innovus floorplan overview.

CHAPTER 3

Advanced architecture development

Once the design of the base filter was carried out, it had to be improved using both:

- Pipelining.
- 3-level unfolding.

This means that there is two ways to proceed: applying first the pipelining and then the unfolding, or vice-versa. Our choice was the second one because in this way the procedure is much more straight for- ward. In figure 3.1a you can see the unfolded filter without pipelining, since this structure is composed by three equal blocks, we proceeded to consider only one of these blocks and to apply pipelining only to this, the result is shown in figure 3.1b. At this point the complete design is given connecting allthe three blocks as shown in figure 3.1c.

The choice to pipeline a block in this way is due to the fact that the first row of registers (green) make the multipliers, that are the combinational blocks with the highest critical path, the only component in them path. Then the other rows of registers divide the adders in groups of three, that is the maximum number of these that can be put in cascade before reaching the critical path of the multiplier (this has been verified performing some test synthesis). So this is the configuration with the minimum critical path, without pipelining internally the components, which at the same time minimise the number of registers and as a consequence the latency.

From here on out, the procedure to simulate, synthesis, and place-and-route the design is equal to the one seen for the basic filter, with the only difference that the testbench is modified to support the unfolding, giving and storing three samples at a time, so here only the final results are reported (refer to appendix C for the complete HDLs, scripts, and reports).

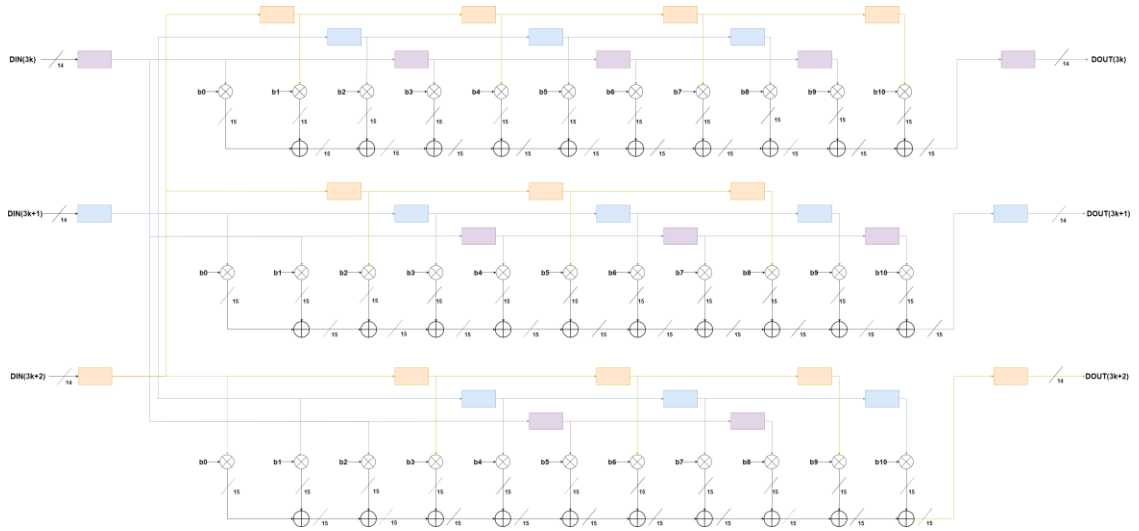
3.1 ModelSim simulation

In figure 3.2a it is possible to see the reset procedure, this time the latency between input signal and output signal is six clock cycles (in place of two) because of the pipeline registers. Furthermore you can notice that now three data are sampled and three data are issued to the output at a time, due to the three level unfolding.

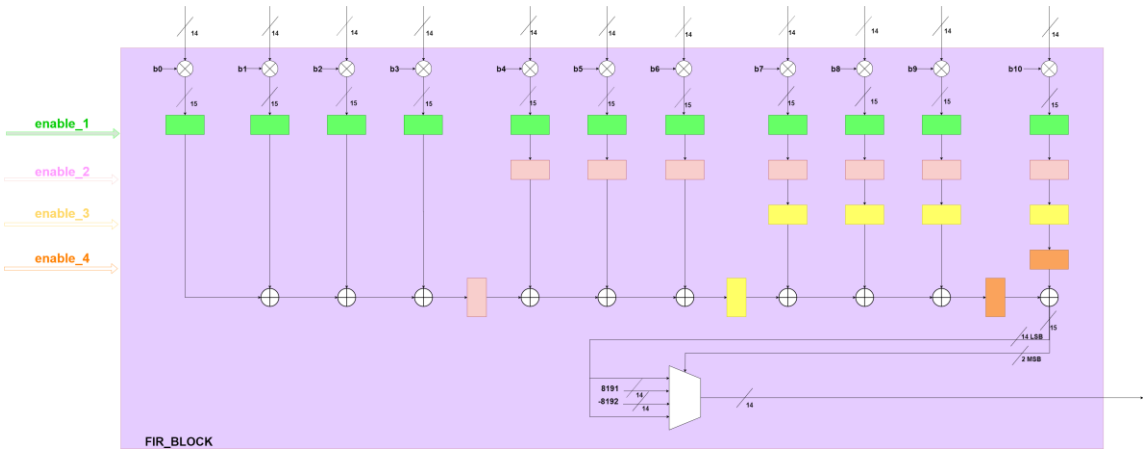
In figure 3.2b you can observe that when *vin* goes to zero, the filter doesn't allow the entering of new samples until *vin* return to one.

In figure 3.2c there is the computation of the last samples and the *ENDSIM* signal behaviour, already explained above.

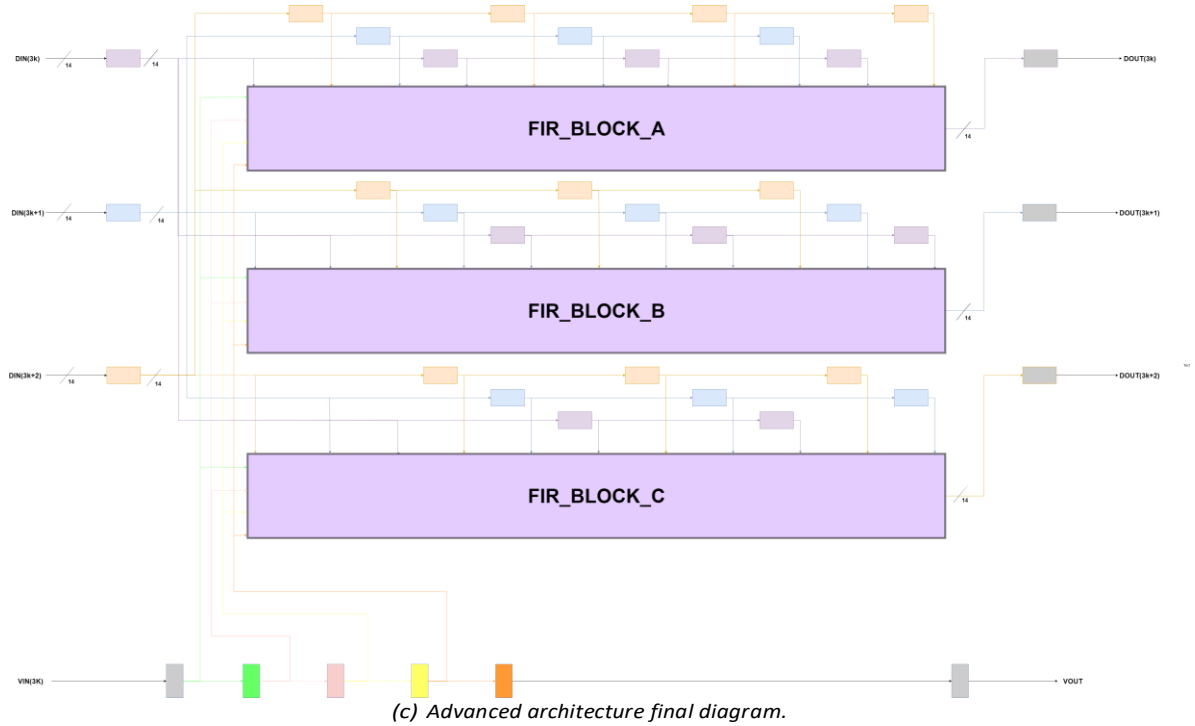
In conclusion, in figure 3.2d the saturations to the maximum and to the minimum value are underlined. In order to test properly this saturation process, the samples provided to the filter are



(a) Unfolded filter.

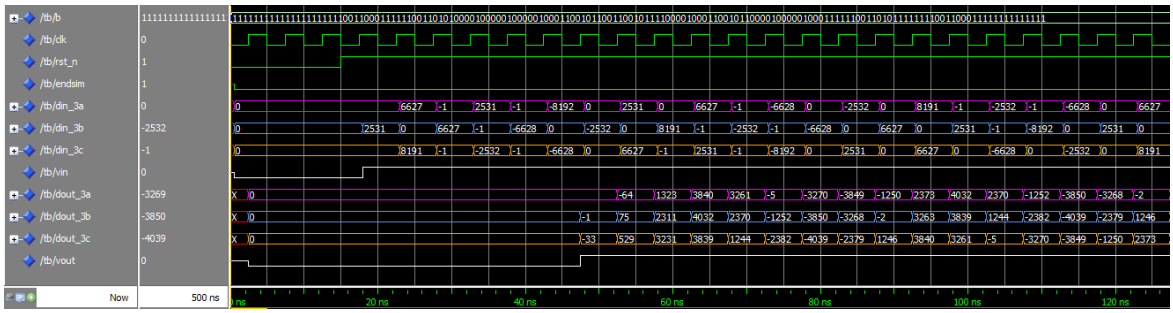


(b) Pipelined block. Different colors are used to underline diffent stages of the pipeline and the corresponding enable signal.

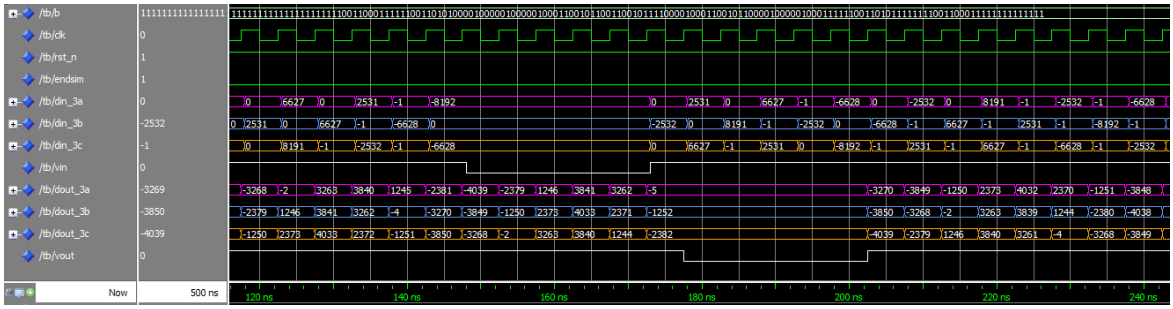


(c) Advanced architecture final diagram.

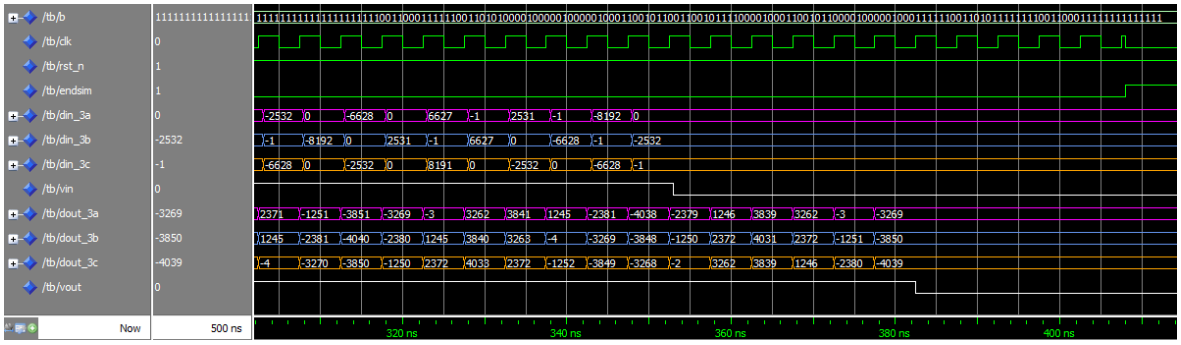
Figure 3.1: Advanced design diagrams. Clock and reset signals are omitted for simplicity. The colors of the pipelining registers are the same of figure 3.1b.



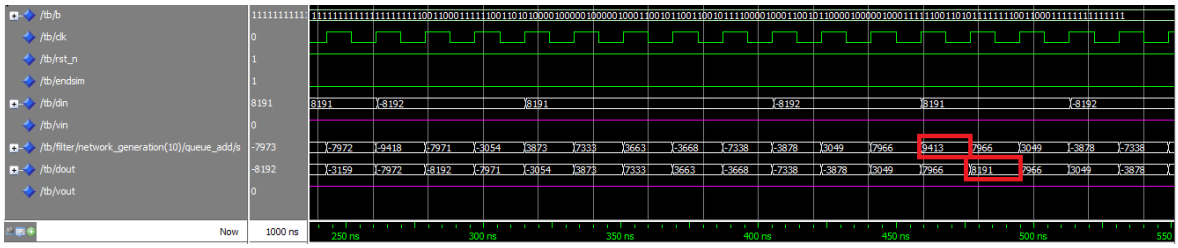
(a) Filter's behaviour at the starting point. Violet refers to data(3k), blue to data(3k+1), and orange to data(3k+2).



(b) Filter's behaviour when the validation input goes to 0: at around 140 ns *it stay down for five clock cycles, at around* 180 ns *all restart normally.*



(c) Filter's behaviour at the end of the process.



(d) Saturation to the minimum at around 60 ns *on the second block, saturation the the maximum at* 80 ns *on the first block, and again saturation to the maximum at* 100 ns *on the third block.*

Figure 3.2: Advanced design ModelSim simulations.

changed with the ones in listing A.5 so that the maximum and the minimum achievable numbers inside the filter, are obtained.

3.2 Synthesis

This time the maximum synthesizable frequency was found equal to 543.48 MHz, so in the final synthesis a frequency of 128.20 MHz is constrained. In tables 3.1 and 3.2 the area and the switching-activity-based power consumption are reported respectively.

Comparing these data with the ones obtained before we can notice some important facts:

- The maximum frequency is more than doubled, this, combined with the unfolding, gives a throughput more than six times bigger respect to the basic design;
- As could have been expected, the area is just over three times larger;
- Also the power consumption is bigger, about 4.4 times.

Max freq. [MHz]	**Comb. area** [μm^2]	**Non comb. area** [μm^2]	**Total area** [μm^2]
543.48	32291.070112	6335.055771	38626.125883
128.20	29094.282230	6331.331770	35425.614000

Table 3.1: Advanced design post synthesis area report.

	Internal Power	**Switching Power**	**Leakage Power**	**Total Power**
mW	3.49	2.42	0.72	6.63
%	52.64	36.50	10.86	100

Table 3.2: Advanced design post synthesis switching-activity-based power consumption.

3.3 Place-and-route

The final area is equal to 35 164.9 μm^2, in figure 3.3 the obtained floorplan in Cadence Innovous graphic envirnoment is shown, while in table 3.3 the post place-and-route power consumption estimated using the switching-activity is reported. Also this time the power consumption is slightly grows from the post-synthesis to the post-place-and-route, for the same reasons explained before.

	Internal Power	**Switching Power**	**Leakage Power**	**Total Power**
mW	5.50	3.59	0.71	9.80
%	56.13	36.43	7.23	100

Table 3.3: Advanced design post place-and-route switching-activity-based power consumption.

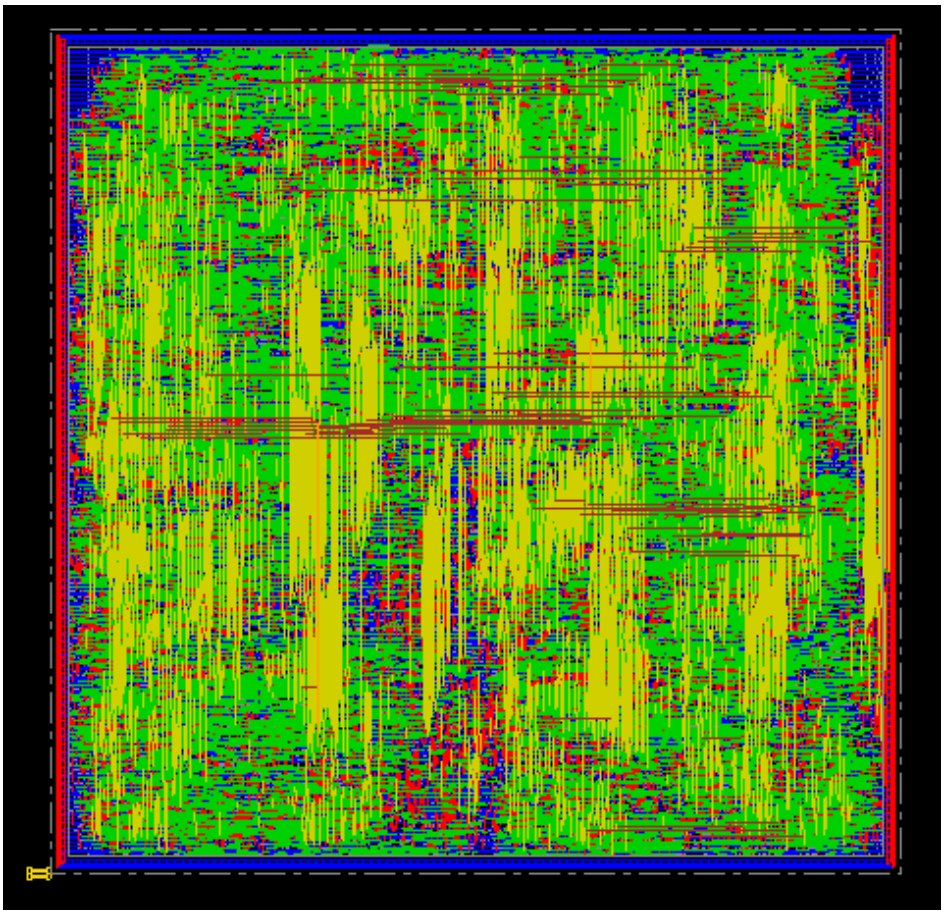


Figure 3.3: Innovous advanced design floorplan overviw.

APPENDIX A

Reference model listings

Listing A.1: myfir_design.m

```
function [bi, bq]=myfir_design(N,nb)
%% function myfir_design(N,nb)
%% N is order of the filter
%% nb is the number of bits
%% bi taps represented as integers

close all;

f_cut_off = 2000; % 2 kHz
f_sampling = 10000; % 10kHz

f_nyq = f_sampling/2; %% Nyquist frequency
f0 = f_cut_off/f_nyq; %% normalized cut-off frequency

b=fir1(N, f0); %% get filter coefficients
[h1, w1]=freqz(b); %% get the transfer function of the designed filter

bi=floor(b*2^(nb-1)); %% convert coefficients into nb-bit integers
bq=bi/2^(nb-1); %% convert back coefficients as nb-bit real values
[h2, w2]=freqz(bq); %% get the transfer function of the quantized filter

%% show the transfer functions
plot(w1/pi, 20*log10(abs(h1)));
hold on;
plot(w2/pi, 20*log10(abs(h2)),'r--');
grid on;
xlabel('Normalized frequency');
ylabel('dB');
```

Listing A.2: my fir filter.m

```
fs=10000 %% sampling frequency
f1 =500; %% first sinewave freq ( in band )
f2=4500; %% second sinnewave freq (out band)

N =10; %% filter order
nb=14; %% number of bits

T =1/ 500; %% maximum period
tt=0:1/fs:10*T; %% time samples

x1=sin(2*pi*f1*tt); %% first sinewave
x2=sin(2*pi*f2*tt); %% second sinewave
```

```
x=(x1+x2)/2;  %%  input  signal

[bi, bq]=myfir_design(N,  nb);  %%  filter  design

y=filter(bq, 1, x); %% apply filter

%% plots
figure
plot(tt,x1,'--d');
hold on
plot(tt,x2,'r--s');
plot(tt,x, 'g--+');
plot(tt, y, 'c--o');

legend('x1', 'x2', 'x', 'y')

%% quantize input and output
xq=floor(x*2^(nb-1));
idx=find(xq==2^(nb-1));   xq(
idx)=2^(nb-1)-1;

yq=floor(y*2^(nb-1));
idy=find(yq==2^(nb-1));
yq(idy)=2^(nb-1)-1;

%% save input and output
fp=fopen('samples.txt','w');
fprintf(fp,'%d\n', xq);
fclose(fp);

fp=fopen('resultsm.txt', 'w');
fprintf(fp, '%d\n', yq);
fclose(fp);
```

Listing A.3: myfilter.c

```
#include<stdio.h>
#include<stdlib.h>

#define NT 11 /// number of coeffs
#define NB 14 /// number of bits

const int b[NT]={-1, -104, -203, 520, 2251, 3260, 2251, 520, -203, -104, -1}; /// b array

/// Perform fixed point filtering assming direct form I
///\param x is the new input sample
///\return the new output sample
int myfilter(int x)
{
  static int sx[NT]; /// x shift register
  static int sy[NT-1]; /// y shift register
  static int first_run = 0; /// for cleaning shift registers
  int i; /// index
  int y; /// output sample

  /// clean the buffers
  if (first_run == 0)
  {
    first_run = 1;
    for (i=0; i<NT; i++)
      sx[i] = 0;
    for (i=0; i<NT-1; i++)
      sy[i] = 0;
  }

  /// shift and insert new sample in x shift register
  for (i=NT-1; i>0; i--)
    sx[i] = sx[i-1];
  sx[0] = x;
```

```
  /// make the convolution
  /// Moving average part
  y = 0;
  for (i=0; i<NT; i++)
    y += (sx[i]*b[i]) >> (NB-1) ;

  /// update the y shift register
  for (i=NT-2; i>0; i--)
    sy[ i] = sy[i -1];
  sy[0] = y;

  return y;
}

int main (int argc, char **argv)
{
  FILE * fp_in ;
  FILE *fp_out;

  int x;
  int y;

  /// check the command line
  if (argc != 3)
  {
    printf("Use: %s <input_file> <output_file>\n", argv[0]);
    exit(1);
  }

  /// open files
  fp_in = fopen(argv[1], "r");
  if (fp_in == NULL)
  {
    printf("Error: cannot open %s\n");
    exit(2);
  }
  fp_out = fopen(argv[2], "w");

  /// get samples and apply filter
  fscanf(fp_in, "%d", &x);
  do{
    y = myfilter ( x);
    fprintf(fp_out,"%d\n", y);
    fscanf(fp_in, "%d", &x);
  } while (!feof(fp_in));

  fclose ( fp_in );
  fclose ( fp_out );

  return 0;

}
```

Listing A.4: samples.txt

```
0
2531
0
6627
0
8191
-1
6627
-1
2531
-1
-2532
-1
-6628
-1
-8192
0
-6628
0
-2532
0
2531
0
6627
0
8191
-1
6627
-1
2531
-1
-2532
-1
-6628
-1
-8192
0
-6628
0
-2532
0
2531
0
6627
0
8191
0
6627
-1
2531
0
-2532
-1
-6628
-1
-8192
0
-6628
0
-2532
0
2531
0
6627
0
8191
0
6627
-1
2531
-1
-2532
-1
-6628
-1
-8192
0
-6628
0
-2532
0
2531
-1
-2532
-1
-6628
-1
-8192
0
-6628
0
-2532
0
2531
0
6627
0
8191
-1
6627
-1
2531
-1
-2532
0
-6628
0
-8192
0
-6628
-1
-2532
-1
2531
0
6627
-1
8191
-1
6627
-1
2531
-1
-2532
-1
-6628
-1
-8192
-1
-6628
-1
-2532
0
2531
-1
6627
0
8191
0
6627
0
2531
-1
-2532
0
-6628
0
-8192
0
-6628
-1
-2532
0
```

```
2531
0
6627
0
8191
-1
6627
-1
2531
0
-2532
-1
-6628
-1
-8192
-1
-6628
0
-2532
0
2531
-1
6627
0
8191
0
6627
0
2531
-1
-2532
-1
-6628
0
-8192
0
-6628
0
-2532
0
2531
0
6627
-1
8191
-1
6627
0
2531
0
-2532
-1
-6628
0
-8192
-1
-6628
0
-2532
-1
```

Listing A.5: samples_sat.txt

```
8191
8191
8191
-8192
-8192
-8192
-8192
-8192
8191
8191
8191
-8192
-8192
-8192
8191
8191
8191
-8192
-8192
-8192
8191
8191
8191
-8192
-8192
-8192
-8192
-8192
8191
8191
8191
```

Listing A.6: resultsm.txt

```
0
-1
-33
-64
76
530
1324
2312
3233
3844
4037
3845
3266
2376
1247
-1
-1248
-2377
-3267
-3846
-4038
-3846
-3267
-2377
-1248
-1
1247
2376
3266
3845
4037
3845
3266
2376
1247
0
-1248
-2377
-3267
```

```
−3846
−4038
−3846
−3267
−2377
−1248
−1
1247
2376
3266
3845
4037
3845
3266
2376
1247
−1
−1248
−2377
−3267
−3846
−4038
−3846
−3267
−2377
−1248
0
1247
2376
3266
3845
4037
3845
3266
2376
1247
0
−1248
−2377
−3267
−3846
−4038
−3846
−3267
−2377
−1248
0
1247
2376
3266
3845
4037
3845
3266
2376
1247
−1
−1248
−2377
−3267
−3846
−4038
−3846
−3267
−2377
−1248
−1
1247
2376
3266
3845
4037
```

```
3845
3266
2376
1247
0
−1248
−2377
−3267
−3846
−4038
−3846
−3267
−2377
−1248
0
1247
2376
3266
3845
4037
3845
3266
2376
1247
−1
−1248
−2377
−3267
−3846
−4038
−3846
−3267
−2377
−1248
−1
1247
2376
3266
3845
4037
3845
3266
2376
1247
0
−1248
−2377
−3267
−3846
−4038
−3846
−3267
−2377
−1248
0
1247
2376
3266
3845
4037
3845
3266
2376
1247
−1
−1248
−2377
−3267
−3846
−4038
−3846
−3267
```

```
−2377
−1248
0
1247
2376
3266
3845
4037
3845
3266
2376
1247
−1
−1248
−2377
−3267
−3846
−4038
```

Listing A.7: resultsc.txt

```
0
−1
−33
−64
75
529
1323
2311
3231
3840
4032
3839
3261
2370
1244
−5
−1252
−2382
−3270
−3850
−4039
−3849
−3268
−2379
−1250
−2
1246
2373
3263
3840
4032
3839
3261
2370
1244
−5
−1252
−2382
−3270
−3850
−4039
−3849
−3268
−2379
−1250
−2
1246
2373
3263
3841
4033
3840
3262
2372
1245
−4
−1251
−2381
−3270
−3850
−4039
−3849
−3268
−2379
−1250
−2
1246
2373
3263
3841
4033
3840
3262
2371
1244
−5
−1252
−2382
−3270
−3850
−4039
−3849
−3268
−2379
−1250
−2
1246
2373
3263
3840
4032
3839
3261
2370
1244
−4
−1251
−2380
−3268
−3848
−4038
−3849
−3269
−2381
−1252
−4
1245
2371
3262
3839
4031
3838
3261
2370
1244
−5
−1252
−2382
−3270
−3851
```

```
-4040
-3851
-3270
-2381
-1251
-4
1245
2372
3262
3840
4033
3841
3263
2372
1245
-3
-1251
-2380
-3268
-3848
-4038
-3849
-3269
-2380
-1251
-3
1246
2373
3263
3840
4032
3839
3261
2371
1245
-4
-1251
-2381
-3270
-3851
-4040
-3850
-3269
-2380
-1250
-3
1245
2372
3262
3840
4033
3841
3263
2372
1245
-4
-1252
-2381
-3269
-3849
-4038
-3848
-3268
-2379
-1250
-2
1246
2372
3262
3839
4031
3839
```

```
3262
2372
1246
-3
-1251
-2380
-3269
-3850
-4039
```

APPENDIX B

Base design listings

B.1 HDL

Listing B.1: fir package.vhd

```
library ieee;
use ieee.std_logic_1164.all;

-- used to define some signal type
package fir_package is

    type wire_reg is array (10 downto 0) of std_logic_vector(13 downto 0);           -- signal
        ↪ used to manage the connection between register
    type wire_mult_add is array (10 downto 0) of std_logic_vector(14 downto 0);      -- signal
        ↪ used to manage the connection between arithmetic part
end package fir_package ;
```

Listing B.2: fir.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.fir_package.all;        -- used to define some signal type

-- FIR filter (direct structure) of order ten with a parallelism of 14 bits.

entity fir is
    port(
        DIN     : in std_logic_vector(13 downto 0);      -- input data
        VIN     : in std_logic;                          -- validation signal, is 1 when it is
            ↪ received a valid data, 0 otherwise.
        RST_n   : in std_logic;                          -- reset signal active low
        CLK     : in std_logic;                          -- clock signal
        b       : in std_logic_vector(153 downto 0);                              -- coefficent of
            ↪ the filter
        DOUT    : out std_logic_vector(13 downto 0);     -- output data
        VOUT    : out std_logic                          -- validation signal, is 1 when it is
            ↪ generated a valid data, 0 otherwise.
    );
end entity fir;

architecture behavioral of fir is

    component reg is
        Generic (N: positive := 1 );                                          --  number of bits
        Port(    D       : In    std_logic_vector(N-1 downto 0);              --  data input
```

```
            Q       : Out   std_logic_vector(N-1 downto 0);          --  data output
            EN      : In    std_logic;                               --  enable active
                ↪ high
            CLK     : In    std_logic;                               --  clock
            RST_n   : In    std_logic                                --  synchronous reset
                ↪  active low
    );
end component reg;

component Mult is
    generic (N:  integer := 8);                                      --  number of bits
    Port (  A:  IN  std_logic_vector(N-1 downto 0);                  --  data input 1
            B:  IN  std_logic_vector(N-1 downto 0);                  --  data input 2
            M:  OUT std_logic_vector(N downto 0)                     --  multiplication's
                ↪ result
        );
end component Mult;

component Adder is
    generic (N:  integer := 8);                                      --  number of bits
    Port (  A:  In  std_logic_vector(N-1 downto 0);                  --  data input 1
            B:  In  std_logic_vector(N-1 downto 0);                  --  data input 2
            S:  Out std_logic_vector(N-1 downto 0)                   --  sum's result
        );
end component Adder;

component MUX_4to1 is
Generic (N: integer:= 1);                                            --number of bits
Port (  IN0, IN1, IN2, IN3  : In    std_logic_vector(N-1 downto 0);  --data inputs
        SEL                 : In    std_logic_vector(1 downto 0);    --selection input
        Y                   : Out   std_logic_vector(N-1 downto 0));  --data output
end component MUX_4to1;

signal connection_reg   : wire_reg;                          -- Used to connect the
    ↪ register in a sequence
signal mult_to_add      : wire_mult_add;                     -- Connect the outputs of the
    ↪  multipliers to the inputs of adders
signal add_to_add       : wire_mult_add;                     -- Connect the output of the
    ↪ previously adder, to one input of the following adder
signal enable_reg       : std_logic;                         -- Is used to propagate the
    ↪ VIN signal in the internal structure
signal sel_signal       : std_logic_vector(1 downto 0);      -- Used to control the SEL
    ↪ signal of the mux
signal out_mux          : std_logic_vector(13 downto 0);     -- Used to connect the output
    ↪  of the saturation stage to the output register for dout

BEGIN
    -- the first mult is place here to simplify the generation of the following stages
    mult0    : mult
    GENERIC MAP ( N => 14 )
    PORT MAP(A => connection_reg(0), B => b(13 downto 0), M => mult_to_add(0)) ;

    -- all the input signals pass through registers, so two registers are placed for the two
        ↪ input signals (DIN,VIN)
    reg_din : reg
    GENERIC MAP ( N => 14 )
    PORT MAP(D => DIN, Q => connection_reg(0), EN => '1' , CLK => CLK, RST_n => RST_n ) ;

    reg_vin : reg
    GENERIC MAP ( N => 1 )
    PORT MAP(D(0) => VIN, Q(0) => enable_reg, EN =>'1' , CLK => CLK, RST_n => RST_n ) ;

    --simple assigment, that connect the output of multiplier to one input of the adder
        ↪ present in the next branch
    add_to_add(0) <= mult_to_add(0);

    -- generate the structure of the filter: the register chain, 10 adders, and the remaining
        ↪  10 multipliers
    network_generation: for x in 1 to 10 generate
        reg_chain    : reg
        GENERIC MAP ( N => 14 )
```

```
            PORT MAP(D => connection_reg(x-1), Q => connection_reg(x), EN => enable_reg , CLK =>
                ↪ CLK, RST_n => RST_n ) ;

            queue_mult : mult
            GENERIC MAP ( N => 14 )
            PORT MAP(A => connection_reg(x), B => b(13+(14*x) downto 14*x), M => mult_to_add(x))
                ↪ ;

            queue_add   : Adder
            GENERIC MAP ( N => 15 )
            PORT MAP(A => add_to_add(x-1), B => mult_to_add(x), S => add_to_add(x) ) ;

        end generate network_generation;

        -- compose the sel signal to control the mux of the saturation stage
        sel_signal <= ( add_to_add(10)(14) & add_to_add(10)(13));

        -- saturation stage, if the number is lower than -8192 or greater than 8191, the number
            ↪ is saturated at the minimum or maximum respectively
        saturation_stage: MUX_4to1
            Generic map(N => 14)
            Port map ( IN0=> add_to_add(10)(13 downto 0), IN1=>"01111111111111",
                       IN2=> "10000000000000", IN3=> add_to_add(10)(13 downto 0),
                       SEL => sel_signal ,Y => out_mux );

        -- all the outputs pass through registers, so two registers are placed for the two output
            ↪ signals (DOUT,VOUT)
        reg_dout    : reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => out_mux, Q => DOUT, EN => enable_reg , CLK => CLK, RST_n => RST_n ) ;

        reg_vout    : reg
        GENERIC MAP ( N => 1 )
        PORT MAP(D(0) =>enable_reg, Q(0) => VOUT, EN => '1' , CLK => CLK, RST_n => RST_n ) ;
end architecture behavioral;
```

Listing B.3: mult.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_signed.all;
use ieee.numeric_std.all;

-- element used to perform multiplication between two input signal
entity Mult is
    generic (N:  integer := 8);                          --  number of bits
    Port (  A:  IN  std_logic_vector(N-1 downto 0);      --  data input 1
            B:  IN  std_logic_vector(N-1 downto 0);      --  data input 2
            M:  OUT std_logic_vector(N downto 0)         --  multiplication's result
            );
end Mult;

architecture BHV of Mult is

signal y    :   std_logic_vector(2*N-1 downto 0);

begin

    y   <=  A*B;                    --perform multiplication
    M   <=  y(2*N-1 downto N-1);    -- take only the most significant part to obtain a result on
        ↪ N+1 bit

end BHV;
```

Listing B.4: adder.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

-- Element used to perform addition between two signal
entity Adder is
    generic (N:  integer := 8);                          --number of bits
    Port (  A:  In  std_logic_vector(N-1 downto 0);      --data input 1
            B:  In  std_logic_vector(N-1 downto 0);      --data input 2
            S:  Out std_logic_vector(N-1 downto 0)       --sum's result
            );
end  Adder;

architecture BHV of Adder is
    signal y    :   std_logic_vector(N downto 0);
    signal a_i  :   std_logic_vector(N downto 0);
    signal b_i  :   std_logic_vector(N downto 0);

 begin

    a_i <= A(N-1) & A;            -- extend signal A
    b_i <= B(N-1) & B;            -- extend signal B
    y <= a_i + b_i;               -- sum
    S <= y(N-1 downto 0);         -- cut the signal to obtain one with the same number of bits of
        ↪ the input signals

end  BHV;
```

Listing B.5: reg.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;

-- register description
entity reg is
    Generic (N: positive:= 1 );                                  --  number of bits
    Port(   D       : In    std_logic_vector(N-1 downto 0);      --  data input
            Q       : Out   std_logic_vector(N-1 downto 0);      --  data output
            EN      : In    std_logic;                           --  enable active high
            CLK     : In    std_logic;                           --  clock
            RST_n   : In    std_logic                            --  synchronous reset active low
    );
end entity reg;

architecture behavioral of reg is    -- register with asyncronous reset and enable

begin
    PSYNCH: process(CLK)             -- synchronous reset
    begin
        if rising_edge(CLK) then     -- otherwise if there is a positive clock edge
            if RST_n='0' then        -- if reset is active
                Q<=(others=>'0');    -- clear the output
            elsif EN='1' then        -- and enable is active
                Q <= D;              -- writes the input on the output
            end if;
        end if;
    end process;

end behavioral;
```

Listing B.6: mux_4to1.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;

entity MUX_4to1 is
    Generic (N: integer:= 1);    --number of bits
    Port (   IN0, IN1, IN2, IN3   : In    std_logic_vector(N-1 downto 0);      --data inputs
             SEL                  : In    std_logic_vector(1 downto 0);       --selection input
             Y                    : Out   std_logic_vector(N-1 downto 0));    --data output
end entity MUX_4to1;

architecture behavioral of MUX_4to1 is
begin

    with SEL select Y<=
        IN0 when "00",
        IN1 when "01",
        IN2 when "10",
        IN3 when others;

end architecture behavioral;
```

B.2 Testbench

Listing B.7: tb_fir.v

```
// `timescale 1 ns

module tb ();

    wire [13:0] din;
    wire vin;
    wire rst_n;
    wire clk;
    wire [153:0] b;
    wire [13:0] dout;
    wire vout;
    wire endsim;

    fir DUT(
        .DIN(din),
        .VIN(vin),
        .RST_n(rst_n),
        .CLK(clk),
        .b(b),
        .DOUT(dout),
        .VOUT(vout)
        );

    data_maker data_maker_inst(
        .CLK(clk),
        .RST_n(rst_n),
        .VOUT(vin),
        .DOUT(din),
        .B(b),
        .END_SIM(endsim)
    );

    data_sink data_sink_inst(
        .CLK(clk),
        .RST_n(rst_n),
        .VIN(vout),
        .DIN(dout)
    );
```

```
    clk_gen   clk_gen_inst (
        .END_SIM(endsim),
        .CLK(clk),
        .RST_n(rst_n)
    );

endmodule
```

Listing B.8: clk_gen.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

-- used to generate the clock and reset signal
entity clk_gen is
  port (
    END_SIM : in  std_logic;
    CLK     : out std_logic;
    RST_n   : out std_logic);
end clk_gen;

architecture beh of clk_gen is

    constant Ts : time := 18 ns;          -- set the period of the clock signal ( value comes from
        ↪ the Fmax*4, obtained after synthesis)

    signal CLK_i : std_logic;

begin  -- beh

    process
    begin  -- process
        if (CLK_i = 'U') then
            CLK_i <= '0';
        else
            CLK_i <= not(CLK_i);
        end if;
        wait for Ts/2;
    end process;

    CLK <= CLK_i and not(END_SIM);

    process
    begin  -- process
        RST_n <= '0';
        wait for 3*Ts;
        RST_n <= '1';
        wait;
    end process;

end beh;
```

Listing B.9: data_maker.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_textio.all;
use ieee.numeric_std.all;

library std;
use std.textio.all;
```

```
-- This component is used to simulate the filter. It pass at every clock cycle one data to the
    ↪ filter and provide the coefficent
-- used by the filter ( in this case the order of the filter is ten, so 11 coefficent are passed)
    ↪ .
entity data_maker is
  port (
    CLK      : in  std_logic;                          -- clock signal
    RST_n    : in  std_logic;                          -- reset signal active low, asynchronous
    VOUT     : out std_logic;                          -- validation signal, when 1 the DOUT is
        ↪ valid, when 0 the process is stopped
    DOUT     : out std_logic_vector(13 downto 0);      -- data in read from an external file which
        ↪ is provide to the filter
    B        : out std_logic_vector(153 downto 0);     -- coefficent pass to the filter
    END_SIM : out std_logic);                          -- notify the end of simulation
end data_maker;

architecture beh of data_maker is

    constant tco     : time := 0.5 ns;                 -- time between clock edge and signal change
    constant N       : integer:= 14;                   -- filter has a parallelism of 14 bits

    signal sEndSim  : std_logic;
    signal END_SIM_i: std_logic_vector(0 to 10);
    signal not_valid: std_logic;                       --if it is = 1, all goes in normal way, if it
        ↪  is 0, the valid signal is put to 0
begin  -- beh

    -- assign the value to the coefficients
    B( 13 downto 0)<= std_logic_vector(to_signed(-1, N));
    B( 27 downto 14)<= std_logic_vector(to_signed(-104, N));
    B( 41 downto 28)<= std_logic_vector(to_signed(-203, N));
    B( 55 downto 42)<= std_logic_vector(to_signed(520, N));
    B( 69 downto 56)<= std_logic_vector(to_signed(2251, N));
    B( 83 downto 70)<= std_logic_vector(to_signed(3260, N));
    B( 97 downto 84)<= std_logic_vector(to_signed(2251, N));
    B( 111 downto 98)<= std_logic_vector(to_signed(520, N));
    B( 125 downto 112)<= std_logic_vector(to_signed(-203, N));
    B( 139 downto 126)<= std_logic_vector(to_signed(-104, N));
    B( 153 downto 140)<= std_logic_vector(to_signed(-1, N));

    -- this process manage the data generation and control the valid signal
    process (CLK, RST_n)
        file fp_in : text open READ_MODE is "samples.txt";      -- put "samples_sat.txt" if you
            ↪ want test the saturation stage
        variable line_in : line;
        variable x : integer;
        variable cc: integer:=0;

        begin  -- process
            if RST_n = '0' then                        -- asynchronous reset (active low)
                DOUT <= (others => '0') after tco;
                VOUT  <=  '0'   after  tco  ;
                s End Sim <= '0' after tco ;
                not_valid  <= '1'  after tco;
            elsif rising_edge(CLK) then                -- rising clock edge

                cc  := cc+1;                           -- count the  number of clock cycle
                if (cc = 25) then
                    not_valid  <= '0';                 -- when arrive at 25, it put the validation
                        ↪ signal to 0
                end if;
                if (cc = 30) then                      -- after 5 clock cycle, it stars at the same
                    ↪ point in which it is stopped
                    not_valid  <= '1';
                end if;

                if not_valid ='1'  then
                    if not endfile ( fp_in ) then
                        readline(fp_in, line_in);
                        read(line_in, x);
                        DOUT <=  conv_std_logic_vector(x, N) after tco;
```

```
                        VOUT  <=  '1'  after  tco;
                        sEndSim  <=  '0'  after  tco;
                    else
                        VOUT  <=  '0'  after  tco;
                        sEndSim <= '1' after tco;
                    end if;
                else
                    VOUT  <=  '0'  after  tco;
                end if;
            end if;
        end process;

    -- process used to stopped the computation after 10 clock cycle from the last read data
    process  (CLK,  RST_n)
        begin  -- process
            if RST_n  =  '0'  then                                  -- asynchronous  reset  (active  low)
                END_SIM_i <=  ( others => '0') after  tco;
            elsif rising_edge(CLK) then                             -- rising  clock  edge
                END_SIM_i(0)  <=  sEndSim  after  tco;
                END_SIM_i(1  to 10)  <=  END_SIM_i(0  to  9)  after  tco;
            end if;
    end process;
    END_SIM  <=  END_SIM_i(10);



end  beh;
```

Listing B.10: data_sink.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

entity data_sink is
  port (
    CLK    : in std_logic;                          -- clock signal
    RST_n  : in std_logic;                          -- asynchronous reset (active low)
    VIN    : in std_logic;                          -- validation signal
    DIN    : in std_logic_vector(13 downto 0));     -- data input provide by the filter
end data_sink;

architecture beh of data_sink is

begin

    process  (CLK,  RST_n)
        file res_fp : text open WRITE_MODE is "./results.txt";
        variable line_out : line;
    begin
        if RST_n  =  '0'  then
            null;
        elsif rising_edge(CLK) then                     -- rising clock edge, if vin=1 write on
            ↪ file, otherwise do nothing
            if  (VIN  =  '1')  then
                write(line_out,  conv_integer(signed(DIN)));
                writeline(res_fp,  line_out);
            end if;
        end if;
    end process;

end  beh;
```

B.3 Synthesis

Listing B.11: syn_script.tcl

```
# ********************************************************************************
#  This  script  is  used  to  synthesize  the  fir  filter,  basic  structure
# ********************************************************************************
#preserve name netlist
set power_preserve_rtl_hier_name  true
#analyze all the used files
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_basic/src/reg.vhd}
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_basic/src/MUX_4to1.vhd}
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_basic/src/Adder.vhd}
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_basic/src/mult.vhd}
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_basic/src/fir_package.vhd}
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_basic/src/fir.vhd}
#elaborate the top entity
elaborate fir -architecture behavioral -library WORK
#****************** CONSTRAINT THE SYNTHESIS (timing) **********************
# create a clock signal to constraint the sequential paths
set WCP 17.8
create_clock -name "CLOCK" -period $WCP CLK
set_dont_touch_network CLOCK
#clock uncertainty
set_clock_uncertainty 0.07 [get_clocks CLOCK]
# verify the correct creation of the clock
report_clock > clock_test.rpt
#input delay
set_input_delay 0.5 -max -clock CLOCK [remove_from_collection [all_inputs] CLK]
#max output delay
set_output_delay 0.5 -max -clock CLOCK [all_outputs]
# output load
set OUT_LOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
set_load $OUT_LOAD [all_outputs]
# compile
compile
# timing, area and power report
report_timing > post_syn_timing.rpt
report_area > post_syn_area.rpt
report_power > post_syn_power.rpt
# #****************** generation file **********************
ungroup -all -flatten
change_names -hierarchy -rules verilog

write_sdf post_syn_fir.sdf
write -f verilog -hierarchy -output post_syn_fir.v
write_sdc post_syn_fir.sdc
```

Listing B.12: post syn area.rpt

```
****************************************
Report : area
Design : fir
Version: O-2018.06-SP4
Date   : Fri Nov  8 17:25:08 2019
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/
        ↪ NangateOpenCellLibrary_typical_ecsm_nowlm.db)

Number of ports:                         2649
Number of nets:                          8331
Number of cells:                         4852
Number of combinational cells:           4613
```

```
Number of sequential cells:                 170
Number of macros/black boxes:                 0
Number of buf/inv:                          581
Number of references:                        36

Combinational area:                9047.990060
Buf/Inv area:                       309.092003
Noncombinational area:              768.739972
Macro/Black Box area:                 0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                   9816.730033
Total area:                 undefined
1
```

Listing B.13: post_syn_power_sa.rpt

```
****************************************
Report : power
        -analysis_effort low
Design : fir
Version: O-2018.06-SP4
Date   : Sat Nov  9 14:32:32 2019
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/
        ↪ NangateOpenCellLibrary_typical_ecsm_nowlm.db)


Operating Conditions: typical    Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design         Wire Load Model            Library
------------------------------------------------
fir                    5K_hvratio_1_1     NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  416.6092 uW   (53%)
  Net Switching Power  =  374.6310 uW   (47%)
                         ---------
Total Dynamic Power    =  791.2402 uW  (100%)

Cell Leakage Power     =  200.7797 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )
    ↪ Attrs
--------------------------------------------------------------------------------------------------
    ↪
io_pad              0.0000           0.0000              0.0000           0.0000  (   0.00%)
memory              0.0000           0.0000              0.0000           0.0000  (   0.00%)
black_box           0.0000           0.0000              0.0000           0.0000  (   0.00%)
clock_network       0.0000           0.0000              0.0000           0.0000  (   0.00%)
register           75.9435          73.9345         1.3456e+04          163.3339  (  16.46%)
sequential          0.0000           0.0000              0.0000           0.0000  (   0.00%)
```

```
combinational     340.6656          300.6949          1.8732e+05          828.6849   (   83.54%)
--------------------------------------------------------------------------------------------------
Total              416.6091  uW      374.6293  uW      2.0078e+05  nW      992.0189  uW
1
```

Listing B.14: post_syn_timing.rpt

```
Information: Updating design information... (UID-85)

****************************************
Report : timing
         -path  full
         -delay  max
         -max_paths  1
Design : fir
Version:  O-2018.06-SP4
Date   : Fri Nov   8 17:25:08 2019
****************************************

Operating Conditions: typical    Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: b[16] ( input port clocked by CLOCK )
  Endpoint: reg_dout/Q_reg[13]
            ( rising edge- triggered flip- flop clocked by CLOCK )
  Path Group: CLOCK
  Path Type: max

  Des/Clust/Port       Wire Load Model        Library
  ------------------------------------------------
  fir                  5K_hvratio_1_1         NangateOpenCellLibrary

  Point                                                          Incr        Path
  ---------------------------------------------------------------------------
  clock CLOCK (rise edge)                                       0.00        0.00
  clock network delay (ideal)                                   0.00        0.00
  input external delay                                          0.50        0.50 r
  b[16] (in)                                                    0.00        0.50 r
  queue_mult_1/B[2] (Mult_N14_10)                               0.00        0.50 r
  queue_mult_1/mult_21/b[2] (Mult_N14_10_DW_mult_tc_0)
                                                                0.00        0.50 r
  queue_mult_1/mult_21/U669/ZN (XNOR2_X1)                       0.07        0.57 r
  queue_mult_1/mult_21/U668/ZN (OAI22_X1)                       0.04        0.61 f
  queue_mult_1/mult_21/U667/ZN (NAND2_X1)                       0.03        0.64 r
  queue_mult_1/mult_21/U665/Z (MUX2_X1)                         0.05        0.69 r
  queue_mult_1/mult_21/U422/ZN (INV_X1)                         0.02        0.71 f
  queue_mult_1/mult_21/U662/ZN (AOI222_X1)                      0.10        0.81 r
  queue_mult_1/mult_21/U421/ZN (INV_X1)                         0.03        0.84 f
  queue_mult_1/mult_21/U661/ZN (AOI222_X1)                      0.09        0.93 r
  queue_mult_1/mult_21/U420/ZN (INV_X1)                         0.03        0.96 f
  queue_mult_1/mult_21/U660/ZN (AOI222_X1)                      0.09        1.05 r
  queue_mult_1/mult_21/U419/ZN (INV_X1)                         0.03        1.08 f
  queue_mult_1/mult_21/U659/ZN (AOI222_X1)                      0.09        1.17 r
  queue_mult_1/mult_21/U416/ZN (INV_X1)                         0.03        1.20 f
  queue_mult_1/mult_21/U658/ZN (AOI222_X1)                      0.09        1.29 r
  queue_mult_1/mult_21/U415/ZN (INV_X1)                         0.03        1.32 f
  queue_mult_1/mult_21/U657/ZN (AOI222_X1)                      0.09        1.41 r
  queue_mult_1/mult_21/U413/ZN (INV_X1)                         0.03        1.44 f
  queue_mult_1/mult_21/U656/ZN (AOI222_X1)                      0.11        1.55 r
  queue_mult_1/mult_21/U655/ZN (OAI222_X1)                      0.07        1.61 f
  queue_mult_1/mult_21/U654/ZN (AOI222_X1)                      0.11        1.73 r
  queue_mult_1/mult_21/U653/ZN (OAI222_X1)                      0.07        1.79 f
  queue_mult_1/mult_21/U17/CO (FA_X1)                           0.10        1.89 f
  queue_mult_1/mult_21/U16/S (FA_X1)                            0.14        2.02 r
  queue_mult_1/mult_21/product[14] (Mult_N14_10_DW_mult_tc_0)
                                                                0.00        2.02 r
  queue_mult_1/M[1] (Mult_N14_10)                               0.00        2.02 r
  queue_add_1/B[1] (Adder_N15_0)                                0.00        2.02 r
```

```
queue_add_1/add_23/B[1] (Adder_N15_0_DW01_add_0)         0.00       2.02 r
queue_add_1/add_23/U1_1/S (FA_X1)                        0.12       2.15 f
queue_add_1/add_23/SUM[1] (Adder_N15_0_DW01_add_0)       0.00       2.15 f
queue_add_1/S[1] (Adder_N15_0)                           0.00       2.15 f
queue_add_2/A[1] (Adder_N15_9)                           0.00       2.15 f
queue_add_2/add_23/A[1] (Adder_N15_9_DW01_add_0)         0.00       2.15 f
queue_add_2/add_23/U1_1/CO (FA_X1)                       0.10       2.25 f
queue_add_2/add_23/U1_2/CO (FA_X1)                       0.09       2.34 f
queue_add_2/add_23/U1_3/CO (FA_X1)                       0.09       2.43 f
queue_add_2/add_23/U1_4/CO (FA_X1)                       0.09       2.52 f
queue_add_2/add_23/U1_5/CO (FA_X1)                       0.09       2.61 f
queue_add_2/add_23/U1_6/S (FA_X1)                        0.14       2.75 r
queue_add_2/add_23/SUM[6] (Adder_N15_9_DW01_add_0)       0.00       2.75 r
queue_add_2/S[6] (Adder_N15_9)                           0.00       2.75 r
queue_add_3/A[6] (Adder_N15_8)                           0.00       2.75 r
queue_add_3/add_23/A[6] (Adder_N15_8_DW01_add_0)         0.00       2.75 r
queue_add_3/add_23/U1_6/S (FA_X1)                        0.12       2.87 f
queue_add_3/add_23/SUM[6] (Adder_N15_8_DW01_add_0)       0.00       2.87 f
queue_add_3/S[6] (Adder_N15_8)                           0.00       2.87 f
queue_add_4/A[6] (Adder_N15_7)                           0.00       2.87 f
queue_add_4/add_23/A[6] (Adder_N15_7_DW01_add_0)         0.00       2.87 f
queue_add_4/add_23/U1_6/CO (FA_X1)                       0.10       2.97 f
queue_add_4/add_23/U1_7/S (FA_X1)                        0.14       3.11 r
queue_add_4/add_23/SUM[7] (Adder_N15_7_DW01_add_0)       0.00       3.11 r
queue_add_4/S[7] (Adder_N15_7)                           0.00       3.11 r
queue_add_5/A[7] (Adder_N15_6)                           0.00       3.11 r
queue_add_5/add_23/A[7] (Adder_N15_6_DW01_add_0)         0.00       3.11 r
queue_add_5/add_23/U1_7/S (FA_X1)                        0.12       3.22 f
queue_add_5/add_23/SUM[7] (Adder_N15_6_DW01_add_0)       0.00       3.22 f
queue_add_5/S[7] (Adder_N15_6)                           0.00       3.22 f
queue_add_6/A[7] (Adder_N15_5)                           0.00       3.22 f
queue_add_6/add_23/A[7] (Adder_N15_5_DW01_add_0)         0.00       3.22 f
queue_add_6/add_23/U1_7/CO (FA_X1)                       0.10       3.33 f
queue_add_6/add_23/U1_8/S (FA_X1)                        0.14       3.46 r
queue_add_6/add_23/SUM[8] (Adder_N15_5_DW01_add_0)       0.00       3.46 r
queue_add_6/S[8] (Adder_N15_5)                           0.00       3.46 r
queue_add_7/A[8] (Adder_N15_4)                           0.00       3.46 r
queue_add_7/add_23/A[8] (Adder_N15_4_DW01_add_0)         0.00       3.46 r
queue_add_7/add_23/U1_8/S (FA_X1)                        0.12       3.58 f
queue_add_7/add_23/SUM[8] (Adder_N15_4_DW01_add_0)       0.00       3.58 f
queue_add_7/S[8] (Adder_N15_4)                           0.00       3.58 f
queue_add_8/A[8] (Adder_N15_3)                           0.00       3.58 f
queue_add_8/add_23/A[8] (Adder_N15_3_DW01_add_0)         0.00       3.58 f
queue_add_8/add_23/U1_8/CO (FA_X1)                       0.10       3.68 f
queue_add_8/add_23/U1_9/S (FA_X1)                        0.14       3.82 r
queue_add_8/add_23/SUM[9] (Adder_N15_3_DW01_add_0)       0.00       3.82 r
queue_add_8/S[9] (Adder_N15_3)                           0.00       3.82 r
queue_add_9/A[9] (Adder_N15_2)                           0.00       3.82 r
queue_add_9/add_23/A[9] (Adder_N15_2_DW01_add_0)         0.00       3.82 r
queue_add_9/add_23/U1_9/S (FA_X1)                        0.12       3.94 f
queue_add_9/add_23/SUM[9] (Adder_N15_2_DW01_add_0)       0.00       3.94 f
queue_add_9/S[9] (Adder_N15_2)                           0.00       3.94 f
queue_add_10/A[9] (Adder_N15_1)                          0.00       3.94 f
queue_add_10/add_23/A[9] (Adder_N15_1_DW01_add_0)        0.00       3.94 f
queue_add_10/add_23/U1_9/CO (FA_X1)                      0.10       4.04 f
queue_add_10/add_23/U1_10/CO (FA_X1)                     0.09       4.13 f
queue_add_10/add_23/U1_11/CO (FA_X1)                     0.09       4.22 f
queue_add_10/add_23/U1_12/CO (FA_X1)                     0.09       4.31 f
queue_add_10/add_23/U1_13/CO (FA_X1)                     0.09       4.40 f
queue_add_10/add_23/U1_14/S (FA_X1)                      0.15       4.55 r
queue_add_10/add_23/SUM[14] (Adder_N15_1_DW01_add_0)
                                                         0.00       4.55 r
queue_add_10/S[14] (Adder_N15_1)                         0.00       4.55 r
saturation_stage/SEL[1] (MUX_4to1_N14)                   0.00       4.55 r
saturation_stage/U4/ZN (AND2_X1)                         0.12       4.67 r
saturation_stage/U21/ZN (AOI22_X1)                       0.06       4.73 f
saturation_stage/U20/ZN (NAND2_X1)                       0.03       4.76 r
saturation_stage/Y[13] (MUX_4to1_N14)                    0.00       4.76 r
reg_dout/D[13] (reg_N14_1)                               0.00       4.76 r
reg_dout/U4/ZN (AOI22_X1)                                0.03       4.79 f
reg_dout/U3/ZN (INV_X1)                                  0.03       4.82 r
reg_dout/Q_reg[13]/D (DFF_X1)                            0.01       4.83 r
```

```
  data arrival time                                                  4.83

  clock CLOCK (rise edge)                                 17.80      17.80
  clock network delay (ideal)                              0.00      17.80
  clock uncertainty                                       -0.07      17.73
  reg_dout/Q_reg[13]/CK (DFF_X1)                           0.00      17.73 r
  library setup time                                      -0.03      17.70
  data required time                                                 17.70
  ---------------------------------------------------------------------------
  data required time                                                 17.70
  data arrival time                                                  -4.83
  ---------------------------------------------------------------------------
  slack (MET)                                                        12.87


1
```

B.4 Place-and-route

Listing B.15: pr_script.cmd

```
#############################################################
#
#  Innovus Command Logging File
#  Created on Sat Nov  9 13:53:32 2019
#
#############################################################

#@(#)CDS: Innovus v17.11-s080_1 (64bit) 08/04/2017 11:13 (Linux 2.6.18-194.el5)
#@(#)CDS: NanoRoute 17.11-s080_1 NR170721-2155/17_11-UB (database version 2.30, 390.7.1) {
    ↪ superthreading v1.44}
#@(#)CDS: AAE 17.11-s034 (64bit) 08/04/2017 (Linux 2.6.18-194.el5)
#@(#)CDS: CTE 17.11-s053_1 () Aug 1 2017 23:31:41 ( )
#@(#)CDS: SYNTECH 17.11-s012_1 () Jul 21 2017 02:29:12 ( )
#@(#)CDS: CPE v17.11-s095
#@(#)CDS: IQRC/TQRC 16.1.1-s215 (64bit) Thu Jul 6 20:18:10 PDT 2017 (Linux 2.6.18-194.el5)

set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default
suppressMessage ENCEXT-2799
getDrawView
loadWorkspace -name Physical
win
set init_design_netlisttype verilog
set init_design_settop 1
set init_top_cell fir
set init_verilog ../netlist/post_syn_fir.v
set init_lef_file /software/dk/nangate45/lef/NangateOpenCellLibrary.lef
set init_gnd_net VSS
set init_pwr_net VDD
set init_mmmc_file mmm_design.tcl
init_design
getIoFlowFlag
setIoFlowFlag 0
floorPlan -coreMarginsBy die -site FreePDK45_38x28_10R_NP_162NW_34O -r 1 0.6 5 5 5 5
uiSetTool select
getIoFlowFlag
fit
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
```

```
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0 -avoid_short 0
    ↪ -skip_crossing_trunks none -stacked_via_top_layer metal10 -stacked_via_bottom_layer
    ↪ metal1 -via_using_exact_crossover_size 1 -orthogonal_only true -skip_via_on_pin {
    ↪ standardcell } -skip_via_on_wire_shape { noshape }
addRing -nets {VDD VSS} -type core_rings -follow core -layer {top metal1 bottom metal1 left
    ↪ metal2 right metal2} -width {top 0.8 bottom 0.8 left 0.8 right 0.8} -spacing {top 0.8
    ↪ bottom 0.8 left 0.8 right 0.8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8} -center 1
    ↪ extend_corner {} - threshold 0 - jog_distance 0 - snap_wire_center_to_grid None
clearGlobalNets
global Net Connect VDD -type pgpin -pin VDD -inst * - module {}
global Net Connect VSS -type pgpin -pin VSS -inst * - module {}
setSrouteMode -viaConnectToShape { noshape }
sroute -connect { blockPin padPin padRing corePin floatingStripe } -layerChangeRange { metal1(1)
    ↪ metal10(10) } -blockPinTarget { nearestTarget } -padPinPortConnect { allPort oneGeom }
    ↪ -padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd } -floatingStripeTarget
    ↪ { blockring padring ring stripe ringpin blockpin followpin } -allowJogging 1
    ↪ -crossoverViaLayerRange { metal1(1) metal10(10) } -nets { VDD VSS } -allowLayerChange 1
    ↪ -blockPin useLef -targetViaLayerRange { metal1(1) metal10(10) }
setPlaceMode -prerouteAsObs {1 2 3 4 5 6 7 8}
setPlaceMode -fp false
placeDesign
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS
optDesign -postCTS -hold
getFillerMode -quiet
addFiller -cell FILLCELL_X8 FILLCELL_X4 FILLCELL_X32 FILLCELL_X2 FILLCELL_X16 FILLCELL_X1 -prefix
    ↪ FILLER
set Nano Route Mode -quiet - timing Engine {}
set Nano Route Mode -quiet - route With Si Post Route Fix 0
setNanoRouteMode -quiet -drouteStartIteration default set
NanoRouteMode -quiet -routeTopRoutingLayer default set
NanoRouteMode -quiet -routeBottomRoutingLayer default set
NanoRouteMode -quiet -drouteEndIteration default set
Nano Route Mode -quiet - route With Timing Driven false set
Nano Route Mode -quiet - route With Si Driven false route
Design -globalDetail
setAnalysisMode -analysisType onChipVariation
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postRoute
opt Design - post Route -hold
save Design fir. enc
reset_parasitics
extractRC
rcOut - setload fir. setload - rc_corner my_rc
rcOut - setres fir. setres - rc_corner my_rc
rcOut -spf fir.spf -rc_corner my_rc
rcOut -spef fir.spef -rc_corner my_rc
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]} > /dev/null
timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths 50 -prefix fir_postRoute
    ↪ -outDir timingReports
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]} > /dev/null
timeDesign -postRoute -hold -pathReports -slackReports -numPaths 50 -prefix fir_postRoute -outDir
    ↪ timingReports
verifyConnectivity -type all -error 1000 -warning 50
setVerifyGeometryMode -area { 0 0 0 0 } -minWidth true -minSpacing true -minArea true -sameNet
    ↪ true -short true -overlap true -offRGrid false -offMGrid true -mergedMGridCheck true
    ↪ -minHole true -implantCheck true -minimumCut true -minStep true -viaEnclosure true
    ↪ -antenna false -insuffMetalOverlap true -pinInBlkg false -diffCellViol true -sameCellViol
    ↪ false -padFillerCellsOverlap true -routingBlkgPinOverlap true -routingCellBlkgOverlap
    ↪ true -regRoutingOnly false -stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
    ↪ false -maxWidth true -maxNonPrefLength -1 -error 1000
verifyGeometry
```

```
setVerifyGeometryMode -area { 0 0 0 0 }
reportGateCount -level 5 -limit 100 -outfile gate_count.rpt
saveNetlist post_pr_fir.v
all_hold_analysis_views
all_setup_analysis_views
write_sdf -ideal_clock_network post_pr_fir.sdf
reset_parasitics
extractRC
rcOut -setload fir.setload -rc_corner my_rc
rcOut -setres fir.setres -rc_corner my_rc
rcOut -spf fir.spf -rc_corner my_rc
rcOut -spef fir.spef -rc_corner my_rc
set_power_analysis_mode -reset
set_power_analysis_mode -method static -corner max -create_binary_db true -write_static_currents
    ↪ true -honor_negative_energy true -ignore_control_signals true
set_power_output_dir -reset
set_power_output_dir ./
set_default_switching_activity -reset
set_default_switching_activity -input_activity 0.2 -period 10.0read_activity_file
-reset
read_activity_file -format VCD -scope /tb/DUT -start {} -end {} -block {} ../vcd/design.vcd
set_power -reset
set_powerup_analysis -reset
set_dynamic_power_simulation -reset
report_power -rail_analysis_format VS -outfile .// post_pr_power_sa.rpt
report power -outfile Report after sw activity -sort { total }
```

Listing B.16: post_pr_area.rpt

```
Gate area 0.7980 um^2
Level 0 Module fir                                          Gates=      12306 Cells=     4808 Area=
    ↪ 9820.2 um^2
```

Listing B.17: post_pr_power_sa.rpt

```
*-----------------------------------------------------------------------------------------
*   Innovus 17.11-s080_1 (64bit) 08/04/2017 11:13 (Linux 2.6.18-194.el5)
*
*
*   Date & Time:     2019-Nov-09 14:15:12 (2019-Nov-09 13:15:12 GMT)
*
*-----------------------------------------------------------------------------------------
*
*   Design: fir
*
*   Liberty Libraries used:
*           MyAnView: /software/dk/nangate45/liberty/
    ↪ NangateOpenCellLibrary_typical_ecsm_nowlm.lib
*
*   Power Domain used:
*       Rail:          VDD     Voltage:         1.1
*
*       Power View : MyAnView
*
*       User-Defined Activity : N.A.
*
*   Switching Activity File used:
*           ../vcd/design.vcd
*           Vcd Window used(Start Time, Stop Time):(1.19121e-18, 1.19158e-18)
*                     Vcd Scale Factor: 1
**                    Design annotation coverage: 0/6044 = 0%
*
*       Hierarchical Global Activity: N.A.
*
*       Global Activity: N.A.
*
```

```
*        Sequential Element Activity: N.A.
*
*        Primary Input Activity: 0.200000
*
*        Default icg ratio: N.A.
*
*        Global Comb ClockGate Ratio: N.A.
*
*    Power Units = 1mW
*
*    Time Units = 1e-09 secs
*
*        report_power -outfile Report_after_sw_activity -sort total
*
-------------------------------------------------------------------------------------


Total Power
-------------------------------------------------------------------------------------
Total Internal Power:         0.72112552          48.2158%
Total Switching Power:        0.57272507          38.2935%
Total Leakage Power:          0.20176906          13.4907%
Total Power:                  1.49561965
-------------------------------------------------------------------------------------


Group                              Internal    Switching      Leakage        Total   Percentage
                                   Power       Power          Power          Power   (%)
-------------------------------------------------------------------------------------
Sequential                          0.08049     0.02733      0.01328       0.1211        8.097
Macro                                     0           0            0            0            0
IO                                        0           0            0            0            0
Combinational                       0.6406      0.5454       0.1885        1.375         91.9
Clock (Combinational)                     0           0            0            0            0
Clock (Sequential)                        0           0            0            0            0
-------------------------------------------------------------------------------------
Total                               0.7211      0.5727       0.2018        1.496          100
-------------------------------------------------------------------------------------


Rail                    Voltage    Internal    Switching      Leakage        Total   Percentage
                                   Power       Power          Power          Power   (%)
-------------------------------------------------------------------------------------
VDD                         1.1     0.7211      0.5727       0.2018        1.496          100


-------------------------------------------------------------------------------------
*    Power Distribution Summary:
*        Highest Average Power:        saturation_stage_U3 (AND2_X1):        0.003978
*        Highest Leakage Power: queue_mult_10_mult_21_U445 (XOR2_X2):      8.373e-05
*        Total Cap:   3.03144e-11 F
*        Total instances in design:   4808
*        Total instances in design with no power:      0
*           Total instances in design with no activity:      0
*        Total Fillers and Decap:      0
-------------------------------------------------------------------------------------
```

APPENDIX C

Advanced design listings

C.1 HDL

Listing C.1: fir_adv_package.vhd

```
library ieee;
use ieee.std_logic_1164.all;

package fir_adv_package is

    type mult_in is array (10 downto 0) of std_logic_vector(13 downto 0);          -- used to
        ↪ put all the signal used as input by the multiplier
    type wire_net is array (10 downto 0) of std_logic_vector (14 downto 0);
    type wire_adder is array (13 downto 0) of std_logic_vector(14 downto 0);       -- used to
        ↪ connect data along the adder

    type wire_delay is array (4 downto 0) of std_logic_vector(13 downto 0);        -- used to
        ↪ collect the data delayed
    type wire_valid_delay is array (4 downto 0) of std_logic;                      -- used to
        ↪ collect all the validation signal delayed

end package fir_adv_package ;
```

Listing C.2: fir_adv.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.fir_adv_package.all; -- used to define some signal type

-- This component implement an advance version of FIR filter. It is applied first the unfolding
    ↪ method with p = 3,
-- and so there are obtain three input : 3k, 3k+1, 3k+2 associated respectively to input A, B and
    ↪ C (same for the output)
-- The three validation signal pass through an and gate, so, it is sufficent that only one goes
    ↪ to 0 that all the
-- structure is blocked.

entity fir_adv is
    port (
        DIN_3A   : in std_logic_vector(13 downto 0);      -- input data A (3k)
        DIN_3B   : in std_logic_vector(13 downto 0);      -- input data B (3k+1)
        DIN_3C   : in std_logic_vector(13 downto 0);      -- input data C (3k+2)
        VIN      : in std_logic;                          -- validation signal for input
        RST_n    : in std_logic;                          -- reset signal, active low, synchronous
        CLK      : in std_logic;                          -- clock signal
```

```
        b       : in std_logic_vector(153 downto 0);      -- eleven coefficent for the filter
        DOUT_3A : out std_logic_vector(13 downto 0);     -- output data A (3k)
        DOUT_3B : out std_logic_vector(13 downto 0);     -- output data B (3k+1)
        DOUT_3C : out std_logic_vector(13 downto 0);     -- output data C (3k+2)
        VOUT    : out std_logic                          -- validation signal for output
    );
end entity fir_adv;

architecture behavioral of fir_adv is

    component reg is
        Generic (N: positive := 1 );                                    --  number of bits
        Port (    D       : In    std_logic_vector(N-1 downto 0);       --  data input
                  Q       : Out   std_logic_vector(N-1 downto 0);       --  data output
                  EN      : In    std_logic;                            --  enable active high
                  CLK     : In    std_logic;                            --  clock
                  RST_n   : In    std_logic                             --  synchronous reset active
                      ↪ low
        );
    end component reg;

    component fir_block is
    port(
        DATA_IN     : in mult_in;                       -- inputs vectors
        enable_line1: in std_logic;                     -- enable signal for pipeline stage 1
        enable_line2: in std_logic;                     -- enable signal for pipeline stage 2
        enable_line3: in std_logic;                     -- enable signal for pipeline stage 3
        enable_line4: in std_logic;                     -- enable signal for pipeline stage 4
        RST_n       : in std_logic;                     -- reset signal
        CLK         : in std_logic;                     -- clock signal
        B           : in std_logic_vector(153 downto 0);                        -- coefficent
        DATA_OUT    : out std_logic_vector(13 downto 0) -- output data
    );
    end component fir_block;

    -- In line_x are put in order the input for the fir_block. line_x(0) feed the first
        ↪ multiplier inside fir block, line_x(10) feed the last one.
    signal line_1       : mult_in;
    signal line_2       : mult_in;
    signal line_3       : mult_in;

    -- are created three delay line in which the input is delayed several times
    -- for example: delay_A(2) stand for: input A pass through two register
    signal delay_A      : wire_delay;
    signal delay_B      : wire_delay;
    signal delay_C      : wire_delay;

    -- delay line to proper manage the enable signal for the different pipeline stages
    signal delay_valid  : wire_valid_delay;

    -- three signal used to connect the output of the fir_block to the output register for DOUT
    signal outA,outB,outC:std_logic_vector(13 downto 0);

    BEGIN

        -- input registers
        reg_in_A    :  reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => DIN_3A, Q =>delay_A(0) , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        reg_in_B    :  reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => DIN_3B, Q =>delay_B(0) , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        reg_in_C    :  reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => DIN_3C, Q =>delay_C(0) , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        reg_in_VIN :  reg
        GENERIC MAP ( N => 1 )
        PORT MAP(D(0) => VIN, Q(0) =>delay_valid(0) , EN => '1' , CLK => CLK, RST_n => RST_n ) ;
```

```
--delay the validation signal with 4 registers
sequence_reg_vin: for x in 0 to 3 generate
    reg_d_vin    : reg
    GENERIC MAP ( N => 1 )
    PORT MAP(D(0) => delay_valid(x), Q(0) => delay_valid(x+1), EN => '1' , CLK => CLK,
        ↪ RST_n => RST_n ) ;
end generate sequence_reg_vin;

--delay the input 3k with a maximum sequence of 3 regs
sequence_reg_A: for x in 0 to 2 generate
    reg_d_A : reg
    GENERIC MAP ( N => 14 )
    PORT MAP(D => delay_A(x), Q => delay_A(x+1), EN => delay_valid(0) , CLK => CLK, RST_n
        ↪ => RST_n ) ;
end generate sequence_reg_A;

--delay the input 3k+1 with a maximum sequence of 3 regs
sequence_reg_B: for x in 0 to 2 generate
    reg_d_B : reg
    GENERIC MAP ( N => 14 )
    PORT MAP(D => delay_B(x), Q => delay_B(x+1), EN => delay_valid(0) , CLK => CLK, RST_n
        ↪ => RST_n ) ;
end generate sequence_reg_B;

--delay the input 3k+2 with a maximum sequence of 3 regs
sequence_reg_C: for x in 0 to 3 generate
    reg_d_C : reg
    GENERIC MAP ( N => 14 )
    PORT MAP(D => delay_C(x), Q => delay_C(x+1), EN => delay_valid(0) , CLK => CLK, RST_n
        ↪ => RST_n ) ;
end generate sequence_reg_C;

-- compose line to assign easly to the fir block. See the comment on the declaration of
    ↪signal delay_x to know how works
    line_1(0) <= delay_A(0);
    line_1 (1) <= delay_C (1) ;
    line_1 (2) <= delay_B (1) ;
    line_1 (3) <= delay_A (1) ;
    line_1 (4) <= delay_C (2) ;
    line_1 (5) <= delay_B (2) ;
    line_1 (6) <= delay_A (2) ;
    line_1 (7) <= delay_C (3) ;
    line_1 (8) <= delay_B (3) ;
    line_1(9)   <= delay_A(3);
    line_1(10)  <= delay_C(4);


    line_2(0)   <= delay_B(0);
    line_2(1)   <= delay_A(0);
    line_2(2)   <= delay_C(1);
    line_2(3)   <= delay_B(1);
    line_2(4)   <= delay_A(1);
    line_2(5)   <= delay_C(2);
    line_2(6)   <= delay_B(2);
    line_2(7)   <= delay_A(2);
    line_2(8)   <= delay_C(3);
    line_2(9)   <= delay_B(3);
    line_2(10)  <= delay_A(3);

    line_3(0)   <= delay_C(0);
    line_3(1)   <= delay_B(0);
    line_3(2)   <= delay_A(0);
    line_3(3)   <= delay_C(1);
    line_3(4)   <= delay_B(1);
    line_3(5)   <= delay_A(1);
    line_3(6)   <= delay_C(2);
    line_3(7)   <= delay_B(2);
    line_3(8)   <= delay_A(2);
    line_3(9)   <= delay_C(3);
    line_3(10)  <= delay_B(3);

-- generate the three block for the advance filter
```

```
        BLOCK_A_FIR : fir_block
        PORT MAP(DATA_IN => line_1 , RST_n => RST_n ,CLK => CLK, B =>b ,DATA_OUT => outA,
                enable_line 1 => delay_valid (0) , enable_line 2 => delay_valid (1) ,
                enable_line3 => delay_valid(2), enable_line4 => delay_valid(3));

        BLOCK_B_FIR : fir_block
        PORT MAP(DATA_IN => line_2 , RST_n => RST_n ,CLK => CLK, B =>b ,DATA_OUT => outB,
                enable_line 1 => delay_valid (0) , enable_line 2 => delay_valid (1) ,
                enable_line3 => delay_valid(2), enable_line4 => delay_valid(3));

        BLOCK_C_FIR : fir_block
        PORT MAP(DATA_IN => line_3 , RST_n => RST_n ,CLK => CLK, B =>b ,DATA_OUT => outC,
                enable_line 1 => delay_valid (0) , enable_line 2 => delay_valid (1) ,
                enable_line3 => delay_valid(2), enable_line4 => delay_valid(3));

        -- output register
        reg_out_A    : reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => outA, Q =>DOUT_3A , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        reg_out_B    : reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => outB, Q =>DOUT_3B , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        reg_out_C    : reg
        GENERIC MAP ( N => 14 )
        PORT MAP(D => outC, Q =>DOUT_3C , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        reg_out_VOUT     : reg
        GENERIC MAP ( N => 1 )
        PORT MAP(D(0) => delay_valid(4), Q(0) =>VOUT , EN => '1' , CLK => CLK, RST_n => RST_n ) ;

        -- simple assignement


end architecture behavioral;
```

Listing C.3: fir_block.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use work.fir_adv_package.all; -- used to define some signal type

-- component in which the structure of the FIR filter is divide in pipeline stages.
entity fir_block is
    port(
        DATA_IN     : in mult_in;                          -- inputs vectors
        enable_line1: in std_logic;                        -- enable signal for pipeline stage 1
        enable_line2: in std_logic;                        -- enable signal for pipeline stage 2
        enable_line3: in std_logic;                        -- enable signal for pipeline stage 3
        enable_line4: in std_logic;                        -- enable signal for pipeline stage 4
        RST_n       : in std_logic;                        -- reset signal
        CLK         : in std_logic;                        -- clock signal
        B           : in std_logic_vector(153 downto 0);   -- coefficent
        DATA_OUT    : out std_logic_vector(13 downto 0)    -- output data
    );
end entity fir_block;

architecture behavioral of fir_block is

    component reg is
        Generic (N: positive:= 1 );                                 --  number of bits
        Port (    D       : In    std_logic_vector(N-1 downto 0);   --  data input
                  Q       : Out   std_logic_vector(N-1 downto 0);   --  data output
                  EN      : In    std_logic;                        --  enable active high
                  CLK     : In    std_logic;                        --  clock
                  RST_n   : In    std_logic                         --  synchronous reset active
                      ↪ low
        );
```

```
end component reg;

component Mult is
    generic (N:  integer := 8);                                    --  number of bits
    Port (  A:  IN  std_logic_vector(N-1 downto 0);                --  data input 1
            B:  IN  std_logic_vector(N-1 downto 0);                --  data input 2
            M:  OUT std_logic_vector(N downto 0)                   --  multiplication's result
        );
end component Mult;

component Adder is
    generic (N:  integer := 8);                                    --  number of bits
    Port (  A:  In  std_logic_vector(N-1 downto 0);                --  data input 1
            B:  In  std_logic_vector(N-1 downto 0);                --  data input 2
            S:  Out std_logic_vector(N-1 downto 0)                 --  sum's result
        );
end component Adder;

component MUX_4to1 is
Generic (N: integer:= 1);    --number of bits
Port (  IN0, IN1, IN2, IN3  : In    std_logic_vector(N-1 downto 0);      --data inputs
        SEL                 : In    std_logic_vector(1 downto 0);        --selection input
        Y                   : Out   std_logic_vector(N-1 downto 0));     --data output
end component MUX_4to1;

signal mult_to_reg      : wire_net;                                    -- connect the output of the
    ↪ multiplier to the input of one register
signal reg_to_adder     : wire_net;                                    -- connect the output of the
    ↪ register to the input of adder
signal reg_to_reg_1     : wire_net;                                    -- connect the output of the
    ↪ register to the input of other register
signal reg_to_reg_2     : wire_net;                                    -- connect the output of the
    ↪ register to the input of other register
signal reg_to_reg_3     : std_logic_vector(14 downto 0);               -- connect the output of the
    ↪ register to the input of other register
signal add_to_adder     : wire_adder;                                  -- connect the output of the
    ↪ adder to the input of other adder
signal sel_signal       : std_logic_vector(1 downto 0);                -- used to control the SEL
    ↪ signal of the mux

BEGIN
    -- place the eleven adder
    network_generation: for x in 0 to 10 generate
        instance_mult    : mult
        GENERIC MAP ( N => 14 )
        PORT MAP(A => DATA_IN(x), B => B(13+(14*x) downto 14*x), M => mult_to_reg(x)) ;
    end generate network_generation;

    -- bank of register to implement pipelininig that divides adder in sequence of 3 max

    pipe_generation_1a: for x in 0 to 3 generate
        reg_pipe 03   : reg
        GENERIC MAP ( N => 15 )
        PORT MAP(D => mult_to_reg(x), Q => reg_to_adder(x), EN => enable_line1 , CLK => CLK,
            ↪ RST_n => RST_n ) ;
    end generate pipe_generation_1a;

    pipe_generation_1b: for x in 4 to 10 generate
        reg_pipe 410 :  reg
        GENERIC MAP ( N => 15 )
        PORT MAP(D => mult_to_reg(x), Q => reg_to_reg_1(x), EN => enable_line1 , CLK => CLK,
            ↪ RST_n => RST_n ) ;
    end generate pipe_generation_1b;

    pipe_generation_2a: for x in 4 to 6 generate
        reg_pipe 46   : reg
        GENERIC MAP ( N => 15 )
        PORT MAP(D => reg_to_reg_1(x), Q => reg_to_adder(x), EN => enable_line2 , CLK => CLK,
            ↪  RST_n => RST_n ) ;
    end generate pipe_generation_ 2 a ;

    pipe_generation_2b: for x in 7 to 10 generate
```

```
    reg_pipe_710 :  reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => reg_to_reg_1(x), Q => reg_to_reg_2(x), EN => enable_line2 , CLK => CLK,
        ↪  RST_n => RST_n ) ;
end generate pipe_generation_2b;

pipe_generation_3a: for x in 7 to 9 generate
    reg_pipe_79   : reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => reg_to_reg_2(x), Q => reg_to_adder(x), EN => enable_line3 , CLK => CLK,
        ↪  RST_n => RST_n ) ;
end generate pipe_generation_3a;

reg_pipe10_1    :  reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => reg_to_reg_2(10), Q => reg_to_reg_3, EN => enable_line3 , CLK => CLK,
        ↪ RST_n => RST_n ) ;

reg_pipe10_2    :  reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => reg_to_reg_3, Q => reg_to_adder(10), EN => enable_line4 , CLK => CLK,
        ↪ RST_n => RST_n ) ;

--   simple   assignement
add_to_adder(0) <= reg_to_adder(0);

-- first three adder
first_chain_adder: for x in 1 to 3 generate
    add_one_to_three    : Adder
    GENERIC MAP ( N => 15 )
    PORT MAP(A => add_to_adder(x-1), B => reg_to_adder(x), S => add_to_adder(x) ) ;
end generate first_chain_adder;

-- first register used to separate the chain of adder
reg_between_1    : reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => add_to_adder(3), Q => add_to_adder(4), EN => enable_line2 , CLK => CLK,
        ↪  RST_n => RST_n ) ;

-- second chain of three adder
second_chain_adder: for x in 4 to 6 generate
    add_four_to_six  : Adder
    GENERIC MAP ( N => 15 )
    PORT MAP(A => add_to_adder(x), B => reg_to_adder(x), S => add_to_adder(x+1) ) ;
end generate second_chain_adder;

-- second register used to separate the chain of adder
reg_between_2    : reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => add_to_adder(7), Q => add_to_adder(8), EN => enable_line3 , CLK => CLK,
        ↪  RST_n => RST_n ) ;

-- third chain of three adder
third_chain_adder: for x in 7 to 9 generate
    add_seven_to_nine    : Adder
    GENERIC MAP ( N => 15 )
    PORT MAP(A => add_to_adder(x+1), B => reg_to_adder(x), S => add_to_adder(x+2) ) ;
end generate third_chain_adder;

-- last register used to separate the chain of adder
reg_between_3    : reg
    GENERIC MAP ( N => 15 )
    PORT MAP(D => add_to_adder(11), Q => add_to_adder(12), EN => enable_line4 , CLK =>
        ↪ CLK, RST_n => RST_n ) ;

-- last adder
add_ten  : Adder
    GENERIC MAP ( N => 15 )
    PORT MAP(A => add_to_adder(12), B => reg_to_adder(10), S => add_to_adder(13) ) ;

-- compose the sel signal to control the mux of the saturation stage
sel_signal <= (add_to_adder(13)(14) & add_to_adder(13)(13));
```

```
        -- saturation stage, if the number is lower than -8192 or greater than 8191, the number
            ↪ is saturated at the minimum or maximum respectively
        saturation_stage: MUX_4to1
            Generic map(N => 14)
            Port map ( IN0=> add_to_adder(13)(13 downto 0), IN1=>"01111111111111",
                       IN2=> "10000000000000", IN3=> add_to_adder(13)(13 downto 0),
                       SEL => sel_signal ,Y =>DATA_OUT );

end architecture behavioral;
```

C.2 Testbench

Listing C.4: tb_adv.v

```
// `timescale 1 ns

module tb ();

    wire [13:0] din_3a;
    wire [13:0] din_3b;
    wire [13:0] din_3c;
    wire vin;
    wire rst_n;
    wire clk;
    wire [153:0] b;
    wire [13:0] dout_3a;
    wire [13:0] dout_3b;
    wire [13:0] dout_3c;
    wire vout;
    wire endsim;

    fir_adv DUT(
        .DIN_3A(din_3a),
        .DIN_3B(din_3b),
        .DIN_3C(din_3c),
        .VIN(vin),
        .RST_n(rst_n),
        .CLK(clk),
        .b(b),
        .DOUT_3A(dout_3a),
        .DOUT_3B(dout_3b),
        .DOUT_3C(dout_3c),
        .VOUT(vout)
        );

    data_maker data_maker_inst(
        .CLK(clk),
        .RST_n(rst_n),
        .VOUT(vin),
        .DOUT_A(din_3a),
        .DOUT_B(din_3b),
        .DOUT_C(din_3c),
        .B(b),
        .END_SIM(endsim)
    );

    data_sink data_sink_inst(
        .CLK(clk),
        .RST_n(rst_n),
        .VIN(vout),
        .DIN_A(dout_3a),
        .DIN_B(dout_3b),
        .DIN_C(dout_3c)
    );
```

```
    clk_gen   clk_gen_inst (
        .END_SIM(endsim),
        .CLK(clk),
        .RST_n(rst_n)
    );

endmodule
```

Listing C.5: data_maker_adv.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_textio.all;
use ieee.numeric_std.all;
library std;
use std.textio.all;

-- This component is used to simulate the filter. It pass at every clock cycle three data to the
    ↪ filter and provide the coefficent
-- used by the filter ( in this case the order of the filter is ten, so 11 coefficent are passed)
    ↪ .
entity data_maker_adv is
  port (
    CLK      : in  std_logic;                          -- clock signal
    RST_n    : in  std_logic;                          -- reset signal
    VOUT     : out std_logic;                          -- validation signal for input
    DOUT_A   : out std_logic_vector(13 downto 0);      -- data generate for input A
    DOUT_B   : out std_logic_vector(13 downto 0);      -- data generate for input B
    DOUT_C   : out std_logic_vector(13 downto 0);      -- data generate for input C
    B        : out std_logic_vector(153 downto 0);     -- coefficent generate
    END_SIM : out std_logic);                          -- signal used to notify the end of the
        ↪ simulation
end data_maker_adv;

architecture beh of data_maker_adv is

    constant tco : time := 0.5 ns;                     -- time between clock edge and signal change
    constant N   : integer:= 14;                    -- filter has a parallelism of 14 bits

    signal sEndSim : std_logic;
    signal END_SIM_i : std_logic_vector(0 to 10);
    signal not_valid: std_logic;                       --if it is = 1, all goes in normal way, if it
        ↪  is 0, the valid signal is put to 0
begin  -- beh

    -- assign the value to the coefficients
    B( 13 downto 0)<= std_logic_vector(to_signed(-1, N));
    B( 27 downto 14)<= std_logic_vector(to_signed(-104, N));
    B( 41 downto 28)<= std_logic_vector(to_signed(-203, N));
    B( 55 downto 42)<= std_logic_vector(to_signed(520, N));
    B( 69 downto 56)<= std_logic_vector(to_signed(2251, N));
    B( 83 downto 70)<= std_logic_vector(to_signed(3260, N));
    B( 97 downto 84)<= std_logic_vector(to_signed(2251, N));
    B( 111 downto 98)<= std_logic_vector(to_signed(520, N));
    B( 125 downto 112)<= std_logic_vector(to_signed(-203, N));
    B( 139 downto 126)<= std_logic_vector(to_signed(-104, N));
    B( 153 downto 140)<= std_logic_vector(to_signed(-1, N));


    -- this process manage the data generation and control the valid signal
    process (CLK, RST_n)
        file fp_in : text open READ_MODE is "samples.txt";           -- put "samples_sat.txt" if
            ↪ you want test the saturation stage
        variable line_in : line;
        variable x,y,z : integer;
        variable cc: integer:=0;
```

```
    begin  -- process
        if RST_n = '0' then                                    -- asynchronous reset (active
            ↪ low)
            VOUT <= '0' after tco;
            DOUT_A <= ( others => '0' ) after tco ;
            DOUT_B <= ( others => '0' ) after tco ;
            DOUT_C <= ( others => '0' ) after tco ;
            sEndSim <= '0' after tco;
            not_valid <= '1' after tco;
        elsif rising_edge(CLK) then  -- rising clock edge

            ------------ TEST VALIDATION SIGNAL ------------
            cc := cc+1;                        -- count the number of clock cycle
            if (cc = 25) then
                not_valid <= '0';              -- when arrive at 25, it put the validation
                    ↪ signal to 0
            end if;
            if (cc = 30) then                  -- after 5 clock cycle, it stars at the same
                ↪ point in which it is stopped
                not_valid <= '1';
            end if;
            ------------------------------------------------

            if not_valid = '1' then

                if not endfile ( fp_in ) then
                    readline(fp_in, line_in);
                    read(line_in, x);
                    DOUT_A <= conv_std_logic_vector(x, N) after tco;
                    VOUT <= '1' after tco;
                    sEndSim <= '0' after tco;
                else
                    VOUT <= '0' after tco ;
                    sEndSim <= '1' after tco;
                end if;

                if not endfile ( fp_in ) then
                    readline(fp_in, line_in);
                    read(line_in, y);
                    DOUT_B <= conv_std_logic_vector(y, N) after tco;
                    VOUT <= '1' after tco;
                    sEndSim <= '0' after tco;
                else
                    VOUT <= '0' after tco ;
                    sEndSim <= '1' after tco;
                end if;

                if not endfile ( fp_in ) then
                    readline(fp_in, line_in);
                    read(line_in, z);
                    DOUT_C <= conv_std_logic_vector(z, N) after tco;
                    VOUT <= '1' after tco;
                    sEndSim <= '0' after tco;
                else
                    VOUT <= '0' after tco ;
                    sEndSim <= '1' after tco;
                end if;
            else
                VOUT <= '0' after tco;
            end if;
        end if;
    end process;

-- process used to stopped the computation after 10 clock cycle from the last read data
process (CLK, RST_n)
    begin  -- process
        if RST_n = '0' then                    -- asynchronous reset (active low)
            END_SIM_i <= (others => '0') after tco;
        elsif rising_edge(CLK) then  -- rising clock edge
            END_SIM_i(0) <= sEndSim after tco;
            END_SIM_i(1 to 10) <= END_SIM_i(0 to 9) after tco;
        end if;
```

```
    end process;

    END_SIM <= END_SIM_i(10);


end beh;
```

Listing C.6: data_sink_adv.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_textio.all;

library std;
use std.textio.all;

-- component used to write the results. It write only when the validation signal is equal to 1
    ↪ for all the three signal(A,B,C)

entity data_sink_adv is
  port (
    CLK     : in std_logic;                          -- clock signal
    RST_n   : in std_logic;                          -- reset signal, active low, asynchronous
    VIN     : in std_logic;                          -- validation signal for input A
    DIN_A   : in std_logic_vector(13 downto 0);      -- input data A (3k)
    DIN_B   : in std_logic_vector(13 downto 0);      -- input data B (3k+1)
    DIN_C   : in std_logic_vector(13 downto 0));     -- input data C (3k+2)
end data_sink_adv;

architecture beh of data_sink_adv is

begin  -- beh

    -- write the result in the file result: if it is not present in the folder of the project, it
        ↪ creates one and write it
    process (CLK, RST_n)
        file res_fp : text open WRITE_MODE is "./results.txt";
        variable line_out : line;
    begin
        if RST_n = '0' then                  -- asynchronous reset (active low)
            null;
        elsif rising_edge(CLK) then          -- rising clock edge
            if (VIN = '1') then
                write(line_out, conv_integer(signed(DIN_A)));
                writeline(res_fp, line_out);
                write(line_out, conv_integer(signed(DIN_B)));
                writeline(res_fp, line_out);
                write(line_out, conv_integer(signed(DIN_C)));
                writeline(res_fp, line_out);
            end if;
        end if;
    end process;

end beh;
```

C.3 Synthesis

Listing C.7: syn_script_adv.tcl

```
# ********************************************************************************
#  This  script  is  used  to  synthesize  the  fir  filter,  advance  structure
# ********************************************************************************
#preserve name netlist
set power_preserve_rtl_hier_name  true
#analyze all the used files
analyze  -library  WORK  -format  vhdl  {/home/isa25/Desktop/lab1/fir_adv/src/reg.vhd}
analyze -library WORK -format vhdl {/home/isa25/Desktop/lab1/fir_adv/src/MUX_4to1.vhd}
analyze -library WORK  -format  vhdl  {/home/isa25/Desktop/lab1/fir_adv/src/Adder.vhd}
analyze  -library  WORK  -format  vhdl  {/home/isa25/Desktop/lab1/fir_adv/src/mult.vhd}
analyze  -library  WORK  -format  vhdl  {/home/isa25/Desktop/lab1/fir_adv/src/fir_adv_package.vhd}
analyze  -library  WORK  -format  vhdl  {/home/isa25/Desktop/lab1/fir_adv/src/fir_block.vhd}
analyze  -library  WORK  -format  vhdl  {/home/isa25/Desktop/lab1/fir_adv/src/fir_adv.vhd}
#elaborate the top entity
elaborate fir_adv -architecture behavioral -library WORK
#****************** CONSTRAINT THE SYNTHESIS (timing) **********************
# create a clock signal to constraint the sequential paths
set WCP 7.8
create_clock -name "CLOCK" -period $WCP CLK
set_dont_touch_network CLOCK
#clock uncertainty
set_clock_uncertainty 0.07 [get_clocks CLOCK]
# verify the correct creation of the clock
report_clock > clock_test.rpt
#input delay
set_input_delay 0.5 -max -clock CLOCK [remove_from_collection [all_inputs] CLK]
#max output delay
set_output_delay 0.5 -max -clock CLOCK [all_outputs]
# output load
set OUT_LOAD [load_of NangateOpenCellLibrary/BUF_X4/A]
set_load $OUT_LOAD [all_outputs]
# compile
compile
#timing and area report
report_timing > post_syn_timing_adv.rpt
report_area > post_syn_area.rpt

# #****************** generation file **********************
ungroup -all -flatten
change_names -hierarchy -rules verilog

write_sdf post_syn_fir_adv.sdf
write -f verilog -hierarchy -output post_syn_fir_adv.v
write_sdc post_syn_fir_adv.sdc
```

Listing C.8: post_syn_area_adv.rpt

```
****************************************
Report : area
Design : fir_adv
Version: O-2018.06-SP4
Date   : Wed Nov 13 11:08:39 2019
****************************************

Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/
        ↪ NangateOpenCellLibrary_typical_ecsm_nowlm.db)

Number of ports:                        10569
Number of nets:                         30655
Number of cells:                        17655
```

```
Number of combinational cells:           15986
Number of sequential cells:               1400
Number of macros/black boxes:                0
Number of buf/inv:                        2884
Number of references:                       26

Combinational area:               29094.282230
Buf/Inv area:                      1539.608015
Noncombinational area:             6331.331770
Macro/Black Box area:                 0.000000
Net Interconnect area:       undefined  (Wire load has zero net area)

Total cell area:                  35425.614000
Total area:                  undefined
1
```

Listing C.9: post syn powe_sa_adv.rpt

```
Information: Updating design information... (UID-85)
Warning: Design 'fir_adv' contains 1 high-fanout nets. A fanout number of 1000 will be used for
    ↪ delay calculations involving these nets. (TIM-134)
Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)
Warning: There is no defined clock in the design. (PWR-80)
Warning: Design has unannotated primary inputs. (PWR-414)
Warning: Design has unannotated sequential cell outputs. (PWR-415)

****************************************
Report : power
        -analysis_effort low
Design : fir_adv
Version: O-2018.06-SP4
Date   : Wed Nov 13 11:01:25 2019
****************************************


Library(s) Used:

    NangateOpenCellLibrary (File: /software/dk/nangate45/synopsys/
        ↪ NangateOpenCellLibrary_typical_ecsm_nowlm.db)


Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design         Wire Load Model            Library
------------------------------------------------
fir_adv                5K_hvratio_1_1     NangateOpenCellLibrary


Global Operating Voltage = 1.1
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =   3.4905 mW    (59%)
  Net Switching Power  =   2.4150 mW    (41%)
Total Dynamic Power    =   5.9055 mW   (100%)

Cell Leakage Power     = 721.4117 uW


                 Internal          Switching           Leakage             Total
Power Group      Power             Power               Power               Power   (   %    )
```

```
-----------------------------------------------------------------------------------------------
    ↪
io_pad              0.0000              0.0000              0.0000              0.0000  (    0.00%)
memory              0.0000              0.0000              0.0000              0.0000  (    0.00%)
black_box           0.0000              0.0000              0.0000              0.0000  (    0.00%)
clock_network       0.0000              0.0000              0.0000              0.0000  (    0.00%)
register        1.3873e+03            356.5840          1.1054e+05          1.8544e+03  (   27.98%)
sequential          0.0000              0.0000              0.0000              0.0000  (    0.00%)
combinational   2.1032e+03          2.0584e+03          6.1087e+05          4.7724e+03  (   72.02%)
-----------------------------------------------------------------------------------------------
    ↪
Total           3.4905e+03 uW       2.4150e+03 uW       7.2141e+05 nW       6.6269e+03 uW
1
```

Listing C.10: post syn timing adv.rpt

```
Information: Updating design information... (UID-85)
Warning: Design 'fir_adv' contains 1 high-fanout nets. A fanout number of 1000 will be used for
    ↪ delay calculations involving these nets. (TIM-134)

****************************************
Report : timing
         -path full
         -delay max
         -max_paths 1
Design : fir_adv
Version: O-2018.06-SP4
Date   : Wed Nov 13 11:08:39 2019
****************************************

 # A fanout number of 1000 was used for high fanout net computations.

Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: reg_in_A/Q_reg[1]
              (rising edge-triggered flip-flop clocked by CLOCK)
  Endpoint: BLOCK_A_FIR/reg_pipe03_0/Q_reg[13]
            (rising edge-triggered flip-flop clocked by CLOCK)
  Path Group: CLOCK
  Path Type: max

  Des/Clust/Port       Wire Load Model        Library
  ------------------------------------------------
  fir_adv              5K_hvratio_1_1         NangateOpenCellLibrary

  Point                                                          Incr        Path
  ---------------------------------------------------------------------------
  clock CLOCK (rise edge)                                        0.00        0.00
  clock network delay (ideal)                                    0.00        0.00
  reg_in_A/Q_reg[1]/CK (DFF_X1)                                  0.00 #      0.00 r
  reg_in_A/Q_reg[1]/Q (DFF_X1)                                   0.21        0.21 r
  reg_in_A/Q[1] (reg_N14_0)                                      0.00        0.21 r
  BLOCK_A_FIR/DATA_IN[0][1] (fir_block_0)                        0.00        0.21 r
  BLOCK_A_FIR/instance_mult_0/A[1] (Mult_N14_0)                  0.00        0.21 r
  BLOCK_A_FIR/instance_mult_0/mult_21/a[1] (Mult_N14_0_DW_mult_tc_0)
                                                                 0.00        0.21 r
  BLOCK_A_FIR/instance_mult_0/mult_21/U443/ZN (INV_X1)
                                                                 0.06        0.27 f
  BLOCK_A_FIR/instance_mult_0/mult_21/U422/ZN (INV_X1)
                                                                 0.12        0.39 r
  BLOCK_A_FIR/instance_mult_0/mult_21/U695/Z (XOR2_X1)
                                                                 0.11        0.50 r
  BLOCK_A_FIR/instance_mult_0/mult_21/U423/ZN (INV_X1)
                                                                 0.07        0.57 f
  BLOCK_A_FIR/instance_mult_0/mult_21/U693/ZN (NAND2_X1)
                                                                 0.12        0.69 r
  BLOCK_A_FIR/instance_mult_0/mult_21/U484/ZN (OAI22_X1)
                                                                 0.06        0.76 f
```

```
BLOCK_A_FIR/instance_mult_0/mult_21/U108/S (HA_X1)       0.08       0.84 f
BLOCK_A_FIR/instance_mult_0/mult_21/U669/ZN (AOI222_X1)
                                                         0.11       0.95 r
BLOCK_A_FIR/instance_mult_0/mult_21/U425/ZN (INV_X1)
                                                         0.03       0.98 f
BLOCK_A_FIR/instance_mult_0/mult_21/U668/ZN (AOI222_X1)
                                                         0.09       1.07 r
BLOCK_A_FIR/instance_mult_0/mult_21/U418/ZN (INV_X1)
                                                         0.03       1.10 f
BLOCK_A_FIR/instance_mult_0/mult_21/U667/ZN (AOI222_X1)
                                                         0.09       1.19 r
BLOCK_A_FIR/instance_mult_0/mult_21/U417/ZN (INV_X1)
                                                         0.03       1.22 f
BLOCK_A_FIR/instance_mult_0/mult_21/U666/ZN (AOI222_X1)
                                                         0.09       1.31 r
BLOCK_A_FIR/instance_mult_0/mult_21/U415/ZN (INV_X1)
                                                         0.03       1.34 f
BLOCK_A_FIR/instance_mult_0/mult_21/U665/ZN (AOI222_X1)
                                                         0.09       1.43 r
BLOCK_A_FIR/instance_mult_0/mult_21/U414/ZN (INV_X1)
                                                         0.03       1.46 f
BLOCK_A_FIR/instance_mult_0/mult_21/U664/ZN (AOI222_X1)
                                                         0.09       1.55 r
BLOCK_A_FIR/instance_mult_0/mult_21/U413/ZN (INV_X1)
                                                         0.03       1.58 f
BLOCK_A_FIR/instance_mult_0/mult_21/U663/ZN (AOI222_X1)
                                                         0.11       1.69 r
BLOCK_A_FIR/instance_mult_0/mult_21/U662/ZN (OAI222_X1)
                                                         0.07       1.76 f
BLOCK_A_FIR/instance_mult_0/mult_21/U661/ZN (AOI222_X1)
                                                         0.11       1.87 r
BLOCK_A_FIR/instance_mult_0/mult_21/U660/ZN (OAI222_X1)
                                                         0.07       1.93 f
BLOCK_A_FIR/instance_mult_0/mult_21/U17/CO (FA_X1)       0.10       2.03 f
BLOCK_A_FIR/instance_mult_0/mult_21/U16/CO (FA_X1)       0.09       2.12 f
BLOCK_A_FIR/instance_mult_0/mult_21/U15/CO (FA_X1)       0.09       2.21 f
BLOCK_A_FIR/instance_mult_0/mult_21/U14/CO (FA_X1)       0.09       2.30 f
BLOCK_A_FIR/instance_mult_0/mult_21/U13/CO (FA_X1)       0.09       2.39 f
BLOCK_A_FIR/instance_mult_0/mult_21/U12/CO (FA_X1)       0.09       2.48 f
BLOCK_A_FIR/instance_mult_0/mult_21/U11/CO (FA_X1)       0.09       2.57 f
BLOCK_A_FIR/instance_mult_0/mult_21/U10/CO (FA_X1)       0.09       2.67 f
BLOCK_A_FIR/instance_mult_0/mult_21/U9/CO (FA_X1)        0.09       2.76 f
BLOCK_A_FIR/instance_mult_0/mult_21/U8/CO (FA_X1)        0.09       2.85 f
BLOCK_A_FIR/instance_mult_0/mult_21/U7/CO (FA_X1)        0.09       2.94 f
BLOCK_A_FIR/instance_mult_0/mult_21/U6/CO (FA_X1)        0.09       3.03 f
BLOCK_A_FIR/instance_mult_0/mult_21/U5/CO (FA_X1)        0.09       3.12 f
BLOCK_A_FIR/instance_mult_0/mult_21/U4/S (FA_X1)         0.13       3.25 r
BLOCK_A_FIR/instance_mult_0/mult_21/product[26] (Mult_N14_0_DW_mult_tc_0)
                                                         0.00       3.25 r
BLOCK_A_FIR/instance_mult_0/M[13] (Mult_N14_0)           0.00       3.25 r
BLOCK_A_FIR/reg_pipe03_0/D[13] (reg_N15_0)               0.00       3.25 r
BLOCK_A_FIR/reg_pipe03_0/U20/ZN (AOI22_X1)               0.03       3.28 f
BLOCK_A_FIR/reg_pipe03_0/U17/ZN (INV_X1)                 0.03       3.31 r
BLOCK_A_FIR/reg_pipe03_0/Q_reg[13]/D (DFF_X1)            0.01       3.31 r
data arrival time                                                   3.31

clock CLOCK (rise edge)                                  7.80       7.80
clock network delay (ideal)                              0.00       7.80
clock uncertainty                                       -0.07       7.73
BLOCK_A_FIR/reg_pipe03_0/Q_reg[13]/CK (DFF_X1)           0.00       7.73 r
library setup time                                      -0.03       7.70
data required time                                                  7.70
--------------------------------------------------------------------------
data required time                                                  7.70
data arrival time                                                  -3.31
--------------------------------------------------------------------------
slack (MET)                                                         4.38


1
```

C.4 Place-and-route

Listing C.11: pr_script_adv.cmd

```
# ######################################################
#
#   Innovus Command Logging File
#   Created on Tue Nov 12 00:06:14 2019
#
# ######################################################

#@(#)CDS: Innovus v17.11-s080_1 (64bit) 08/04/2017 11:13 (Linux 2.6.18-194.el5)
#@(#)CDS: NanoRoute 17.11-s080_1 NR170721-2155/17_11-UB (database version 2.30, 390.7.1) {
     ↪ superthreading v1.44}
#@(#)CDS: AAE 17.11-s034 (64bit) 08/04/2017 (Linux 2.6.18-194.el5)
#@(#)CDS: CTE 17.11-s053_1 () Aug 1 2017 23:31:41 ( )
#@(#)CDS: SYNTECH 17.11-s012_1 () Jul 21 2017 02:29:12 ( )
#@(#)CDS: CPE v17.11-s095
#@(#)CDS: IQRC/TQRC 16.1.1-s215 (64bit) Thu Jul 6 20:18:10 PDT 2017 (Linux 2.6.18-194.el5)

set_global _enable_mmmc_by_default_flow      $CTE::mmmc_default
suppressMessage ENCEXT-2799
getDrawView
loadWorkspace -name Physical
win
set init_design_netlisttype verilog
set init_design_settop 1
set init_top_cell fir_adv
set init_verilog ../netlist/post_syn_fir_adv.v
set init_lef_file /software/dk/nangate45/lef/NangateOpenCellLibrary.lef
set init_gnd_net VSS
set init_pwr_net VDD
set init_mmmc_file mmm_design.tcl
init_design
getIoFlowFlag
setIoFlowFlag 0
floorPlan -coreMarginsBy die -site FreePDK45_38x28_10R_NP_162NW_34O -r 1 0.6 5 5 5 5
uiSetTool select
getIoFlowFlag
fit
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
set sprCreateIeRingLayers {}
set sprCreateIeStripeWidth 10.0
set sprCreateIeStripeThreshold 1.0
setAddRingMode -ring_target default -extend_over_row 0 -ignore_rows 0 -avoid_short 0
     ↪ -skip_crossing_trunks none -stacked_via_top_layer metal10 -stacked_via_bottom_layer
     ↪ metal1 -via_using_exact_crossover_size 1 -orthogonal_only true -skip_via_on_pin {
     ↪ standardcell } -skip_via_on_wire_shape { noshape }
addRing -nets {VDD VSS} -type core_rings -follow core -layer {top metal1 bottom metal1 left
     ↪ metal2 right metal2} -width {top 0.8 bottom 0.8 left 0.8 right 0.8} -spacing {top 0.8
     ↪ bottom 0.8 left 0.8 right 0.8} -offset {top 1.8 bottom 1.8 left 1.8 right 1.8} -center 1
     ↪ extend_corner {} - threshold 0 - jog_distance 0 - snap_wire_center_to_grid None
clearGlobalNets
global Net Connect VDD -type pgpin -pin VDD -inst * - module {}
global Net Connect VSS -type pgpin -pin VSS -inst * - module {}
setSrouteMode -viaConnectToShape { noshape }
```

```
sroute -connect { blockPin padPin padRing corePin floatingStripe } -layerChangeRange { metal1(1)
    ↪ metal10(10) } -blockPinTarget { nearestTarget } -padPinPortConnect { allPort oneGeom }
    ↪ -padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd } -floatingStripeTarget
    ↪ { blockring padring ring stripe ringpin blockpin followpin } -allowJogging 1
    ↪ -crossoverViaLayerRange { metal1(1) metal10(10) } -nets { VDD VSS } -allowLayerChange 1
    ↪ -blockPin useLef -targetViaLayerRange { metal1(1) metal10(10) }
setPlaceMode -prerouteAsObs {1 2 3 4 5 6 7 8}
setPlaceMode -fp false
placeDesign
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postCTS
optDesign -postCTS -hold
getFillerMode -quiet
addFiller -cell FILLCELL_X8 FILLCELL_X4 FILLCELL_X32 FILLCELL_X2 FILLCELL_X16 FILLCELL_X1 -prefix
    ↪ FILLER
set Nano Route Mode -quiet - timing Engine {}
set Nano Route Mode -quiet - route With Si Post Route Fix 0
setNanoRouteMode -quiet -drouteStartIteration default set
NanoRouteMode -quiet -routeTopRoutingLayer default set
NanoRouteMode -quiet -routeBottomRoutingLayer default set
NanoRouteMode -quiet -drouteEndIteration default set
Nano Route Mode -quiet - route With Timing Driven false set
Nano Route Mode -quiet - route With Si Driven false route
Design -globalDetail
setAnalysisMode -analysisType onChipVariation
setOptMode -fixCap true -fixTran true -fixFanoutLoad false
optDesign -postRoute
opt Design - post Route -hold
save Design fir. enc
reset_parasitics
extractRC
rcOut - setload fir. setload - rc_corner my_rc
rcOut - setres fir. setres - rc_corner my_rc
rcOut -spf fir.spf -rc_corner my_rc
rcOut -spef fir.spef -rc_corner my_rc
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]} > /dev/null
timeDesign -postRoute -pathReports -drvReports -slackReports -numPaths 50 -prefix fir_postRoute
    ↪ -outDir timingReports
redirect -quiet {set honorDomain [getAnalysisMode -honorClockDomains]} > /dev/null
timeDesign -postRoute -hold -pathReports -slackReports -numPaths 50 -prefix fir_postRoute -outDir
    ↪ timingReports
verifyConnectivity -type all -error 1000 -warning 50
setVerifyGeometryMode -area { 0 0 0 0 } -minWidth true -minSpacing true -minArea true -sameNet
    ↪ true -short true -overlap true -offRGrid false -offMGrid true -mergedMGridCheck true
    ↪ -minHole true -implantCheck true -minimumCut true -minStep true -viaEnclosure true
    ↪ -antenna false -insuffMetalOverlap true -pinInBlkg false -diffCellViol true -sameCellViol
    ↪ false -padFillerCellsOverlap true -routingBlkgPinOverlap true -routingCellBlkgOverlap
    ↪ true -regRoutingOnly false -stackedViasOnRegNet false -wireExt true -useNonDefaultSpacing
    ↪ false -maxWidth true -maxNonPrefLength -1 -error 1000
verifyGeometry
setVerifyGeometryMode -area { 0 0 0 0 }
reportGateCount -level 5 -limit 100 -outfile post_pr_area_adv.rpt
saveNetlist post_pr_fir_adv.v
all_hold_analysis_views
all_setup_analysis_views
write_sdf -ideal_clock_network post_pr_fir_adv.sdf
reset_parasitics
extractRC
rcOut - setload fir. setload - rc_corner my_rc
rcOut - setres fir. setres - rc_corner my_rc
rcOut -spf fir.spf -rc_corner my_rc
rcOut -spef fir.spef -rc_corner my_rc
set_power_analysis_mode -reset
set_power_analysis_mode -method static -corner max -create_binary_db true -write_static_currents
    ↪ true -honor_negative_energy true -ignore_control_signals true
set_power_output_dir -reset
set_power_output_dir ./
set_default_switching_activity -reset
set_default_switching_activity -input_activity 0.2 -period 10.0
read_activity_file -reset
read_activity_file -format VCD -scope /tb/DUT -start {} -end {} -block {} ../vcd/design.vcd
set_power -reset
```

```
set_powerup_analysis -reset
set_dynamic_power_simulation -reset
report_power -rail_analysis_format VS -outfile .//post_pr_power_sa_adv.rptreport_power
-outfile Report_after_sw_activity -sort { total }
```

Listing C.12: post pr area adv.rpt

```
Gate area 0.7980 um^2
Level 0 Module fir_adv                                    Gates=     44066 Cells=    16970 Area=
    ↪ 35164.9 um^2
```

Listing C.13: post pr power sa adv.rpt

```
*-------------------------------------------------------------------------------------------
*   Innovus 17.11-s080_1 (64bit) 08/04/2017 11:13 (Linux 2.6.18-194.el5)
*
*
*   Date & Time:    2019-Nov-12 15:12:25 (2019-Nov-12 14:12:25 GMT)
*
*-------------------------------------------------------------------------------------------
*
*   Design: fir_adv
*
*   Liberty Libraries used:
*          MyAnView: /home/isa25/Desktop/lab1/fir_adv/innovus/fir_adv.end.dat/libs/mmmc/
    ↪ NangateOpenCellLibrary_typical_ecsm_nowlm.lib
*
*   Power Domain used:
*       Rail:         VDD    Voltage:        1.1
*
*       Power View  :  MyAnView
*
*       User-Defined Activity  : N.A.
*
*   Switching Activity File used:
*          ../vcd/design.vcd
*          Vcd Window used(Start Time, Stop Time):(1017.8, 1017.79)
*                    Vcd Scale Factor: 1
**                   Design annotation coverage: 0/20364 = 0%
*
*       Hierarchical Global Activity: N.A.
*
*       Global Activity: N.A.
*
*       Sequential Element Activity: N.A.
*
*       Primary Input Activity: 0.200000
*
*       Default icg ratio: N.A.
*
*       Global Comb ClockGate Ratio: N.A.
*
*   Power Units = 1mW
*
*   Time Units = 1e-09 secs
*
*       report power -outfile report sw act -sort total
*
-------------------------------------------------------------------------------------------


Total Power
-------------------------------------------------------------------------------------------

Total Internal Power:         5.49943019          56.1302%
Total Switching Power:        3.59023698          36.6439%
Total Leakage Power:          0.70796422           7.2259%
```

```
Total  Power:                   9.79763137
--------------------------------------------------------------------------------


Group                              Internal    Switching      Leakage        Total   Percentage
                                   Power       Power          Power          Power   (%)
--------------------------------------------------------------------------------
Sequential                            1.887       0.4076       0.1101         2.405        24.54
Macro                                     0            0            0             0           0
IO                                        0            0            0             0           0
Combinational                         3.613        3.183       0.5979         7.393        75.46
Clock  (Combinational)                    0            0            0             0           0
Clock  (Sequential)                       0            0            0             0           0
--------------------------------------------------------------------------------
Total                                 5.499         3.59        0.708         9.798         100
--------------------------------------------------------------------------------


Rail                     Voltage    Internal    Switching      Leakage        Total   Percentage
                                    Power       Power          Power          Power   (%)
--------------------------------------------------------------------------------
VDD                          1.1       5.499         3.59        0.708         9.798         100


--------------------------------------------------------------------------------
*   Power Distribution Summary:
*       Highest Average Power: BLOCK_B_FIR_add_one_to_three_3_add_23_U1_12 (FA_X1):        0
     ↪ .003574
*       Highest Leakage Power:       reg_d_vin_1_Q_reg_0_ (DFF_X2):   0.0001112
*       Total Cap:   1.03803e-10 F
*       Total instances in design: 16970
*       Total instances in design with no power:      0
*          Total instances in design with no activity:      0
*       Total Fillers and Decap:      0
--------------------------------------------------------------------------------
```